Universidad Politécnica de Madrid

Facultad de Informática

Trabajo fin de carrera

Interfaz de BaDELE3000

Autor

Adrián González Izquierdo

Tutora

María del Socorro Bernardos Galindo

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El objetivo del presente Trabajo de Fin de Carrera es el estudio, diseño e implementación de una aplicación web que sirva de interfaz para una base de datos relacional llamada *BaDELE3000* que trabaja actualmente con unos 3000 lexemas y sigue en fase de desarrollo. *BaDELE* modela conceptos y principios lingüísticos más o menos complejos, dentro del marco de la Teoría Sentido-Texto. El desarrollo de esta interfaz, permitirá a personas autorizadas, seguir nutriendo la base de datos.

La memoria está formada por cinco capítulos, una bibliografía y un apéndice. Cada capítulo se divide en secciones.

La sección, «Estado de la cuestión», introduce las bases linguísticas en las que se basa el modelo desarrollado, y su situación actual. En la tercera sección, «Especificación de Requisitos Software», se recogen todos los requisitos que el cliente ha transmitido al desarrollador. La siguiente sección de la memoria, «Desarrollo de la aplicación», es la más extensa. Empieza de manera teórica, describiendo los ciclos de vida en cascada y en espiral. En los siguientes apartados de la sección se explican las diferentes fases del ciclo

de vida: fase de análisis, fase de diseño, centrándose en la elaboración del modelo de datos y relacional, fase de implementación, explicando las decisiones importantes en el desarrollo de la aplicación y la estructura modular de la aplicación, y por último, se trata la fase de pruebas, describiendo los tipos de pruebas ejecutados sobre el sistema y cómo se realizaron. En la quinta sección, se añaden las conclusiones obtenidas como fruto del trabajo realizado y se tratan posibles aspectos de mejora en la aplicación. En última sección, se incluye toda la bibliografía consultada y el manual de instalación y usuario de la aplicación.

Capítulo 2

Estado de la cuestión

2.1. Introducción

Se denomina colocación a las unidades fraseológicas formadas por dos unidades léxicas en relación sintáctica. Esta relación es una combinación «restringida» de palabras, porque se trata de casos donde una palabra no se puede combinar con cualquier otra. A la palabra cuyo significado es autónomo, se le llama «base», mientras que la palabra cuya selección es restringida se denomina «colocativo». Por ejemplo, para expresar el sentido de «intenso», se utiliza diferentes colocativos dependiendo de la base. En el caso de que se utilice la base «odio», se dice «mortal», pero en el caso de utilizar «miedo», se utiliza «atroz».

Las colocaciones son una parte fundamental de las lenguas, ya que son utilizadas por los nativos en acciones cotidianas. El estudio de estas combinaciones es fundamental si se quiere dominar verdaderamente una lengua.

Entre las teorías lingüísticas que abordan el estudio de las colocaciones se

encuentra la Teoría Sentido-Texto[3][1]. Esta teoría describe las colocaciones mediante lo que llama funciones léxicas, que se explican más detalladamente en la sección 2.2. La base de datos empleada en este trabajo sirvió por un lado para formalizar computacionalmente este concepto y, por otro, para producir un recurso léxico para español con este tipo de información.

2.2. Funciones léxicas

La función léxica FL es un tipo de función que relaciona una unidad léxica L con otras unidades léxicas L_i que pueden conservar o no su propio sentido[2][2]. Se entiende unidad léxica, como palabras o palabras que tienen significado propio. Por tanto, en el ámbito de las colocaciones, la función léxica sirve para realizar un estudio formal de colocación, describiendo la relación entre la «base» y la «colocación». En el ejemplo de la sección anterior, se usaba la base «miedo». Si se utiliza la función léxica «Magn» que intensifica la base, se obtendrían los valores: abrumador, atroz, feroz, espantoso, etc.

Cuando un valor de una función léxica se aplica a un grupo de unidades léxicas, se denomina glosa de la función léxica. Por ejemplo, «llevar» es una glosa de $Real_1$ ya que combina con todas las prendas y complementos. A menudo, se utiliza «llevar una falda», «llevar unos zapatos». En cambio, «calzar» es un valor específico para ciertas unidades léxicas como «calzar unos zapatos», «calzar unas botas».

[1]Mel'cuk, Igor A. Meaning-Text Models: A recent trend in Soviet linguistics, Annual Review of Anthropology 10: 2762. 1981

[2]María A. Barrios. El dominio de las funciones léxicas en el marco de la Teoría Sentido-Texto, 49

Las funciones léxicas, resultan una herramienta útil tanto para la enseñanza y aprendizaje de lenguas extranjeras, como para la traducción de textos. Por ejemplo, si se toma la colocación inglesa *to pay attention*, con la función léxica «Oper» que describe la relación entre *pay* y *attention*, se tiene que en español:

```
Oper(atención) = otorgar, prestar, poner, etc.
```

En relación a las colocaciones, Mel'cuk y Wanner en 1996, propusieron la idea del Principio de Herencia Léxica. Para ello demostraron que es posible, crear una entrada genérica y otra no genérica para «emoción». La entrada no genérica agrupa las colocaciones de la unidad léxica «emoción», mientras que la genérica agrupa las funciones léxicas que comparten los valores de una serie de lexemas de «emoción». Por ejemplo, la entrada genérica tendría, al menos, la función léxica Oper_1, cuyo valor «sentir» es común a todos los nombres o lexemas de «emoción»: sentir alegría, odio, felicidad, etc.

2.3. Recursos léxicos computacionales dentro del marco de la Teoría Sentido-Texto

En esta sección se van a introduccir, sin profundizar demasiado en aspectos lingüísticos, los diferentes recursos léxicos que se han realizado combinando los campos computacional y el de la Teoría Sentido-Texto.

2.3.1. *DiCo*. Diccionario de la combinatoria francesa

DiCo es una base de datos léxica del francés, desarrollada por el Observatorio de lingüística Sentido-Texto de la Universidad de Montréal, dirigida por Alain Polguère con la colaboración de Igor Mel'cuk. Desde 2005, existe una interfaz web de uso público llamada *DiCouèbe*, desde la que se pueden realizar consultas a *DiCo*. Su dirección web es `http://www.olst.umontreal.ca/dicouebe`.

El elemento principal de esta base de datos o diccionario son las funciones léxicas, tanto las paradigmáticas como las sintagmáticas. No se incluyen definiciones, en su lugar, se insertan etiquetas semánticas y formas proposionales. Se describe cada unidad léxica en lenguaje formal de modo que pueda ser objeto de tratamiento automático[14].

El contenido de *DiCo* es muy restringida, se basa en la descripción de unas unidades léxicas que cumplan:

- Son unidades léxicas comunes del léxico francés.
- Controlan un cierto número de derivaciones semánticas o colocaciones.
- Forman una especie de núcleo léxico de la lengua.

El diseño de *DiCo* permite la elaboración automática de diccionarios para aplicaciones de procesamiento del lenguaje. También es útil para elaborar diccionarios dirigidos al público en general, como el *Lexique actif du fraçais (2007)*[3] para el aprendizaje del vocabulario con unas 20.000 derivaciones semánticas y colocaciones.

[3]http://olst.ling.umontreal.ca/laf/about/

La interfaz web de *DiCo*, *DiCouèbe*, tiene dos modos: experto y estándar, véase Figura 2.1. En el modo estándar se da acceso a las áreas centrales de la estructura léxica de *DiCo*. El formulario consta de varias opciones, divididas en diferentes secciones: lexía, funciones léxicas, locuciones y ejemplos. En el modo experto, además de las secciones anteriores, existen otras: actantes semánticos, actantes sintácticos e información del artículo o entrada. Cada una de las opciones del formulario, tienen una ayuda contextual explicando el significado.

Una vez realizada una consulta, se muestra una tabla en la parte inferior del formulario. Las columnas de la tabla de salida, serán las opciones que haya seleccionado el usuario. Si se realiza una consulta muy genérica, la aplicación genera una tabla demasiado grande y el navegador web quedará bloqueado durante varios segundos. Los datos de las tablas de salida se pueden exportar a un fichero de texto plano con extensión *tab*.

A modo de ejemplo, se ha realizado una consulta sobre *DiCouèbe*, buscando las funciones léxicas y valores de la etiqueta semántica «sentimiento», *sentiment* en francés. La interfaz de consulta de *DiCouèbe* es la que aparece en la Figura 2.1. Una vez realizada la consulta, *DiCo* encuentra 2491 resultados, una parte de estos resultados se muestran en la Figura 2.2.

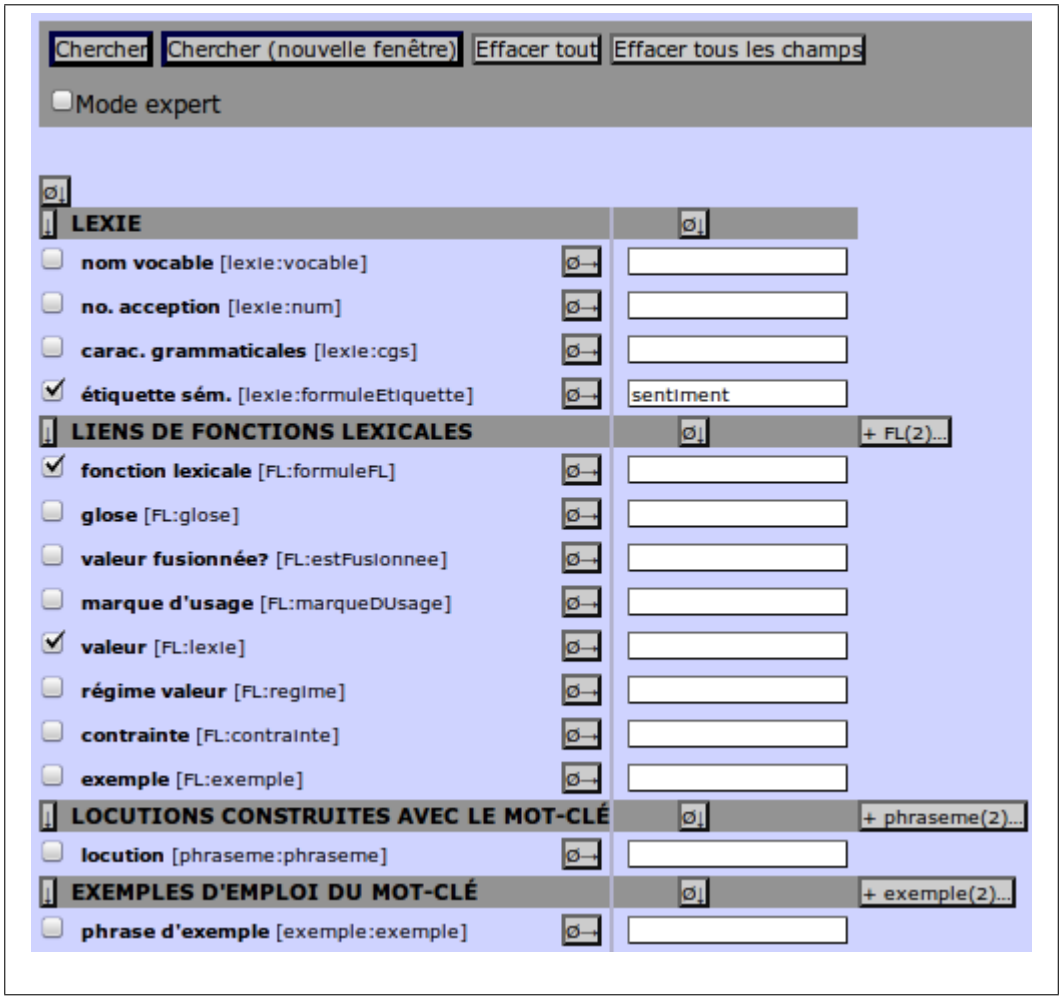
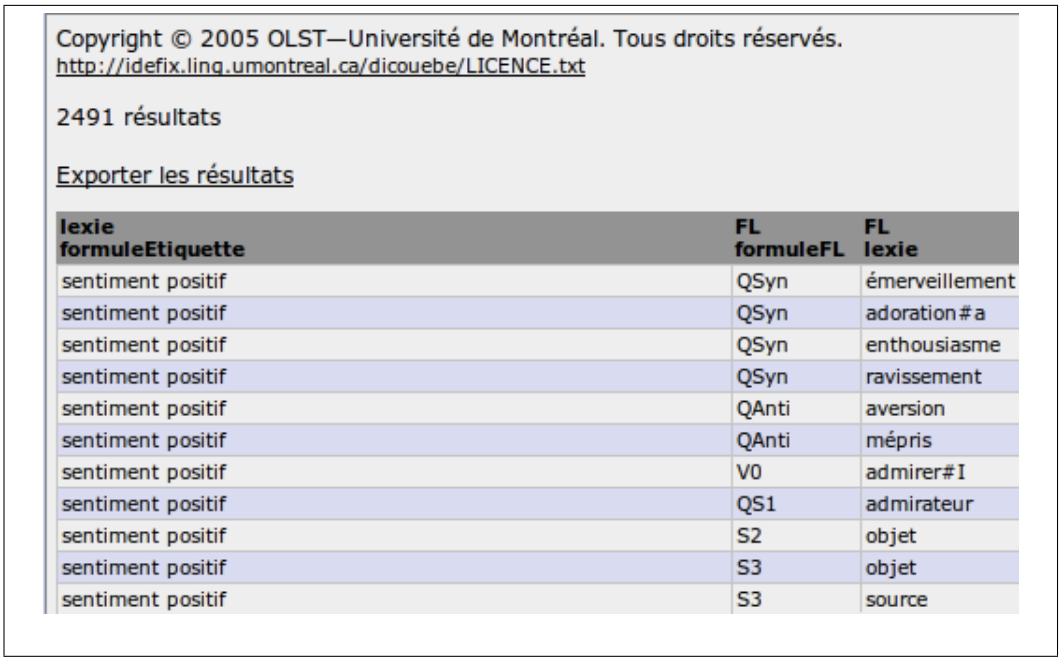


Figura 2.1: *Interfaz de consulta de DiCouèbe modo estándar*



Copyright © 2005 OLST—Université de Montréal. Tous droits réservés.
http://idefix.ling.umontreal.ca/dicouebe/LICENCE.txt

2491 résultats

Exporter les résultats

lexie formuleEtiquette	FL formuleFL	FL lexie
sentiment positif	QSyn	émerveillement
sentiment positif	QSyn	adoration#a
sentiment positif	QSyn	enthousiasme
sentiment positif	QSyn	ravissement
sentiment positif	QAnti	aversion
sentiment positif	QAnti	mépris
sentiment positif	V0	admirer#I
sentiment positif	QS1	admirateur
sentiment positif	S2	objet
sentiment positif	S3	objet
sentiment positif	S3	source

Figura 2.2: *Resultado devuelto por DiCouèbe para sentiment*

2.3.2. *Dice*. Diccionario de colocaciones del Español

Dice es un diccionario de colocaciones del Español desarrollado en la Universidad de A Coruña bajo la dirección de Margarita Alonso Ramos. El ojetivo principal del *DiCe* es proporcionar recursos en línea para el aprendizaje de colocaciones en español. La versión web de este diccionario se puede visitar en la dirección: `http://www.dicesp.com` y cuenta con una serie de ejercicios para el aprendizaje de las colocaciones.

El marco teórico del *DiCe* se basa en la lexicografía explicativa y combinatoria (Mel'uk et al. 1995) y en el diccionario *Lexique actif du fraçais* nombrado en la sección 2.3.1. Por tanto, la función léxica es fundamental al utilizarse como herramienta para representar las relaciones entre unidades léxicas.

El contenido del *DiCe* se organiza en campos semánticos, y actualmente tiene unas 20.000 relaciones léxicas. Hasta el momento, sólo trabaja con el campo semántico de los nombres de «sentimiento», aunque sí obtiene todas las acepciones del lema. Al igual que ocurre con *DiCo*, el *DiCe* no incluye definiciones, en su lugar, se insertan etiquetas semánticas y formas proposicionales[2][4].

El *DiCE* está formado por tres componentes fundamentales: el propio diccionario, consultas avanzadas que permiten navegar por el corpus[5] del diccionario y un módulo didáctico que se encuentra en fase de desarrollo. Si se realiza una consulta por el propio diccionario, Figura 2.3, se ve que muestra información sobre:

- Etiqueta semántica relacionada con el lema introducido en la consulta.
- Forma proposicional, en la que aparecen los participantes de la situación designada por el nombre.
- Ejemplos de uso.
- Los (cuasi-)sinónimos y (cuasi-)antónimos, que ayudarán al usuario a seleccionar el sentido.
- El esquema de régimen, en donde se describe la estructura argumental del nombre.
- Colocaciones y Derivados semánticos.

[4]María A. Barrios. El dominio de las funciones léxicas en el marco de la Teoría Sentido-Texto, 12

[5]Conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación. `http://buscon.rae.es/draeI/`

Cuando se pulsa en una de las colocaciones, por ejemplo «reparo + adjetivo», se obtiene información sobre la función léxica y la glosa relacionada, véase Figura 2.4. En el módulo de consultas avanzadas, se pueden realizar tres tipos de consultas:

- Directas. Sirven para ir de la función léxica a los valores
- Inversas. Permite buscar la base de la colocación a partir del colocativo y también, buscar la función léxica que se combina con una base y un colocativo.
- Ayuda a la redacción. Permite buscar si una combinación dada es correcta o no, véase Figura 2.5.

En comparación con *DiCouèbe*, *DiCe* muestra mucha menos información técnica. En gran parte porque su finalidad es distinta. Es una aplicación que pretende ayudar al aprendizaje del español para personas extranjeras. Por este motivo, a diferencia de *DiCouèbe*, los formularios tienen pocas opciones, los datos de salida no se muestran en formato tabla, resultando más didácticos, y siempre van acompañados de varios ejemplos.

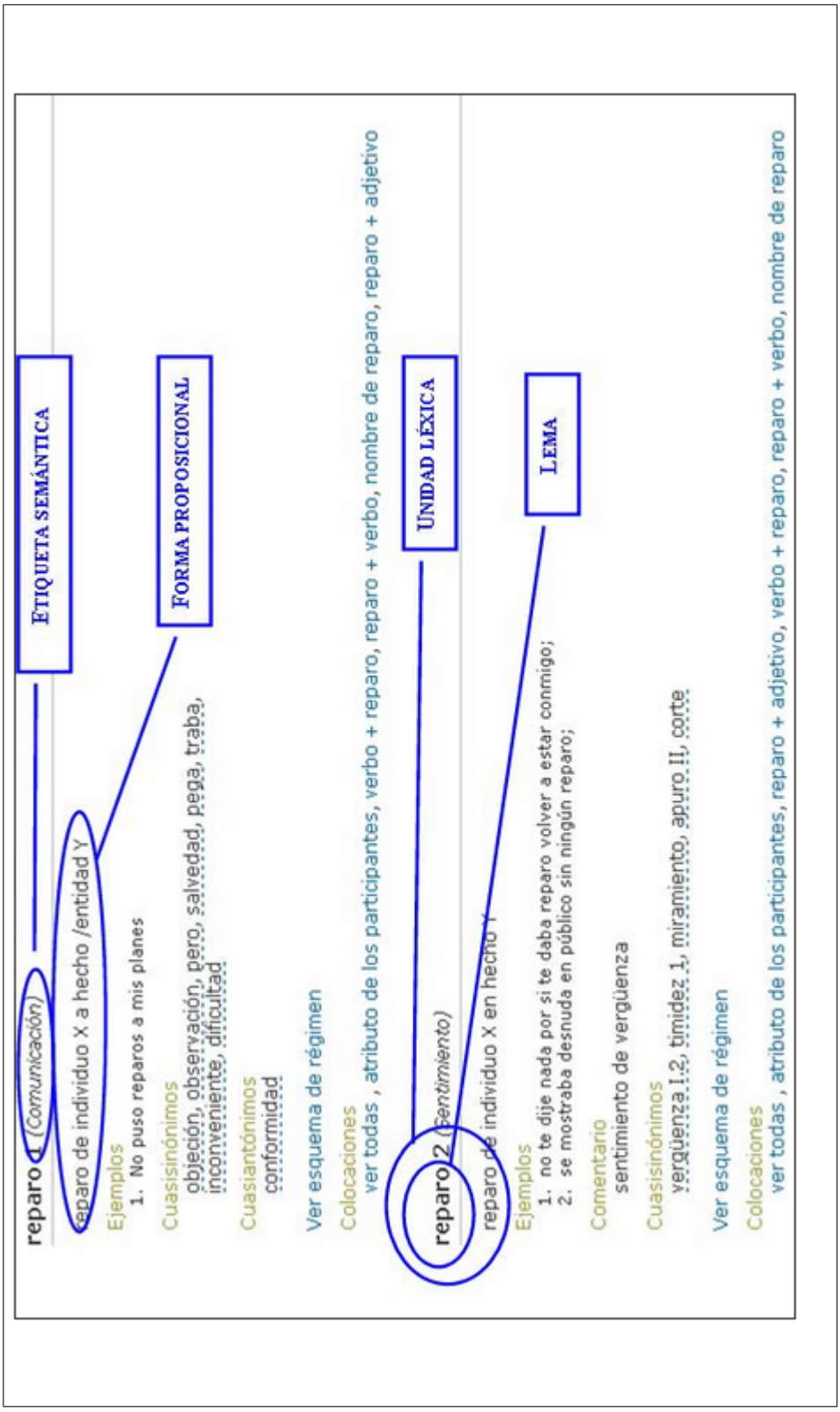


Figura 2.3: *Consulta al diccionario DiCe para el lema «reparo»*

Figura 2.4: *Detalle de la colocación «reparo + adjetivo» en DiCe*

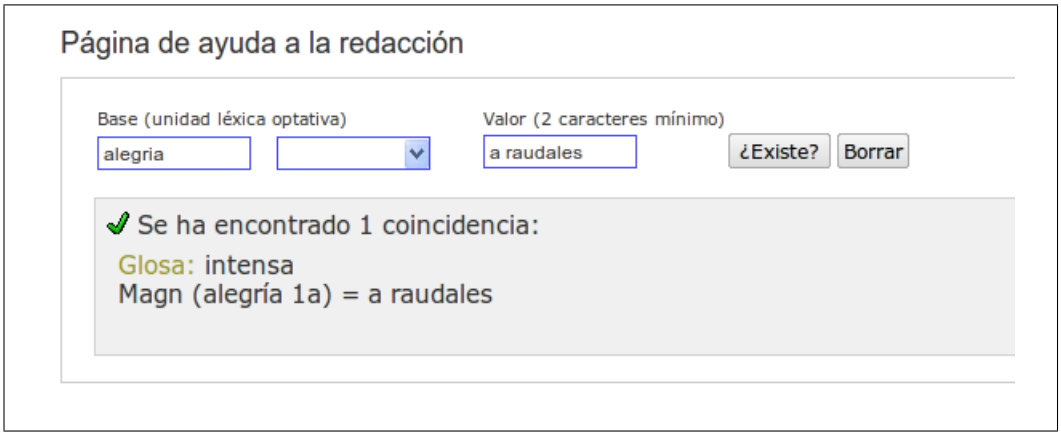


Figura 2.5: *Ejemplo de consulta avanzada de ayuda a la redacción*

2.3.3. CALLEX y su versión española, CALLEX-ESP

CALLEX[6] es un programa multimedia del Instituto de Problemas de Transferencia de Información de la Academia de Ciencias de Moscú para el aprendizaje de ruso, inglés y alemán. Sus autores principales son Jury Apresjan e Igor Boguslavsky. Aunque este programa no está directamente vinculado a la lexicología combinatoria y explicativa[1][7], por el uso de funciones léxicas se considera dentro del marco de la Teoría Sentido-Texto. El sistema es capaz de mostrar las similitudes y diferencias semánticas y sintácticas entre las funciones léxicas de la misma familia y su relación con lexemas que pertenecen al mismo campo léxico.

CALLEX permite el conocimiento léxico al tener la capacidad de generar una serie de actividades divididas en tres niveles y en cada uno de los niveles ocho tipos de ejercicios que el propio sistema corrige. La base de *CALLEX* son unas 2000 palabras de las que se estudian todas las relaciones sintagmáticas y paradigmáticas.

El proyecto de *CALLEX* para el español se denomina *CALLEX-ESP* y se encuentra en fase de desarrollo. Actualmente, contiene los 3.500 sustantivos más utilizados del español, y todos los verbos, adjetivos y adverbios con los que cada uno de ellos guarda algún tipo de relación léxica.

[6] Apresjan et al., 2007a, 2007b, 2003a, 2003b

[7] Barrios, María A., Bernardos, M. Socorro. BaDELE3000. An implementation of the lexical inheritance principle, 1

2.3.4. *BaDELE3000*. Base de Datos para el Español como Lengua Extranjera

BaDELE3000 es una base de datos que contiene unos 3300 sustantivos más usados del español de España, un léxico que se consideró suficiente para el aprendizaje del español en niveles intermedio o avanzado. A partir de estos sustantivos, la base de datos permitió la obtención automática de unas 9.000 relaciones léxicas. En total se obtuvieron unas 20.700 relaciones léxicas con la ayuda de diccionarios combinatorios del español. Para la herencia de glosas y valores de funciones léxicas de cada dominio, elaboraron una clasificación de sustantivos que no existía hasta la fecha[2][8].

El diseño de *BaDELE3000* siguió un proceso de desarrollo sistemático para la construcción de un modelo y base de datos relacionales, intentando asegurar que el acceso a la información de la base de datos fuese lo más sencilla posible. Se utilizaron las funciones léxicas y una implementación del Principio de Herencia Léxica[9], mencionado en la sección 2.2, para que la inserción de información fuese lo más automática posible.

Como ya se ha mencionado, *BaDELE3000* es el origen de este trabajo. Por eso sus detalles se irán explicando a lo largo de los capítulos que restan. Aquí sólo se señalarán los aspectos relevantes con respecto a los otros recursos descritos en este capítulo. Una de las diferencias entre *BaDELE3000* y *DiCo*, reside en la jerarquía de etiquetas semánticas. *DiCo* se basa en un estudio conceptual de las etiquetas semánticas para la clasificación de las mismas,

[8]María A. Barrios. El dominio de las funciones léxicas en el marco de la Teoría Sentido-Texto, 434

[9]Mel'cuk and Wanner, 1996; Mel'cuk, 1996: 76-78

mientras que *BaDELE3000* realiza un estudio estrictamente lingüístico. Otra diferencia, reside en el número de funciones léxicas utilizadas por cada base de datos. En el caso de *BaDELE3000* se han utilizado menos funciones léxicas en comparación con las utilizadas por *DiCo*, ya que esta última herramienta además de las funciones léxicas estándar, utilizadas por *BaDELE3000*, se han utilizado funciones léxicas no estándar, que tienen un sentido específico no generalizable a todas las lenguas. Por último, otra de las diferencias se debe a la propuesta por parte de *BaDELE3000* para automatizar lo máximo posible la obtención de relaciones léxicas reduciendo el tiempo utilizado. En el caso de *DiCo* la automatización es nula.

BaDELE3000 a diferencia de *DiCe* no se centra en un único campo semántico. Por otra parte, *DiCe* hace hincapié en la labor de aprendizaje del español, mostrando múltiples ejemplos y ejercicios.

El diseño de *BaDELE3000* permite que distintas aplicaciones puedan utilizar su contenido, por ejemplo, aplicaciones de traducción automática y generación de texto.

A diferencia del resto de trabajos, *BaDELE3000* no tiene una interfaz amigable ni se puede consultar a través de la web. De ahí el planteamiento de este proyecto.

Capítulo 3

Desarrollo de la aplicación

3.1. Introducción

El desarrollo de una aplicación informática de cierto nivel requiere una serie de fases previas a la implementación. El enfoque clásico propone un ciclo de vida en cascada, véase Figura 3.1, para el desarrollo de software, en el que las fases se siguen de forma sistematica y secuencial.

La primera fase que hay que llevar a cabo es la fase de análisis o educción de requisitos y a continuación se debe realizar una fase de diseño. Posteriormente, en la fase de implementación se obtendrá el resultado deseado para la aplicación al plasmar las ideas y conceptos desarrollados en las fases anteriores. Finalmente, la fase de pruebas permite determinar si la aplicación funciona como se espera y detectar posibles errores cometidos en las fases anteriores, principalmente en la fase de implementación.

Durante la fase de análisis se identifican los requerimientos y necesidades que tiene que satisfacer el sistema. Es una de las tareas más importantes en

el proceso de desarrollo de una aplicación, ya que para comenzar a resolver un problema se debe definir de forma clara y precisa. Se trata de definir el problema, no de proponer una solución al mismo.

En la fase de diseño se analiza el modo en el que se van a implementar los requisitos identificados en la fase anterior. Para ello, se construyen una serie de modelos que detallan el sistema de forma que se lleve fácilmente a la implementación, es decir, a la codificación en el lenguaje de programación escogido. Por tanto, el diseño de una aplicación propone una solución conceptual que satisface los requisitos y es independiente del lenguaje de programación que se utilice en la implementación.

La fase de implementación es en la que se codifica la funcionalidad identificada en la fase de análisis siguiendo las pautas indicadas en la fase de diseño. Cuanto mayor sea la calidad de la solución conceptual propuesta en la fase de diseño, más fácil y sencilla resultará la fase de implementación. Esto no evita que, en la mayoría de los casos, se detecten en la fase de implementación aspectos que no se han tenido en cuenta en las fases anteriores, lo que conlleva realizar modificaciones en los resultados obtenidos en dichas fases.

Finalmente, el objetivo de la fase de pruebas es garantizar que la aplicación desarrollada cubre la funcionalidad requerida y con unas condiciones de calidad. Los posibles errores detectados en la fase de pruebas provocan modificaciones en las fases anteriores, de forma que se reanuda el desarrollo hasta obtener los resultados deseados.

Al ser todas las etapas consecutivas e independientes, hacen que sea un ciclo de vida muy rígido y no admita cambios de manera sencilla.

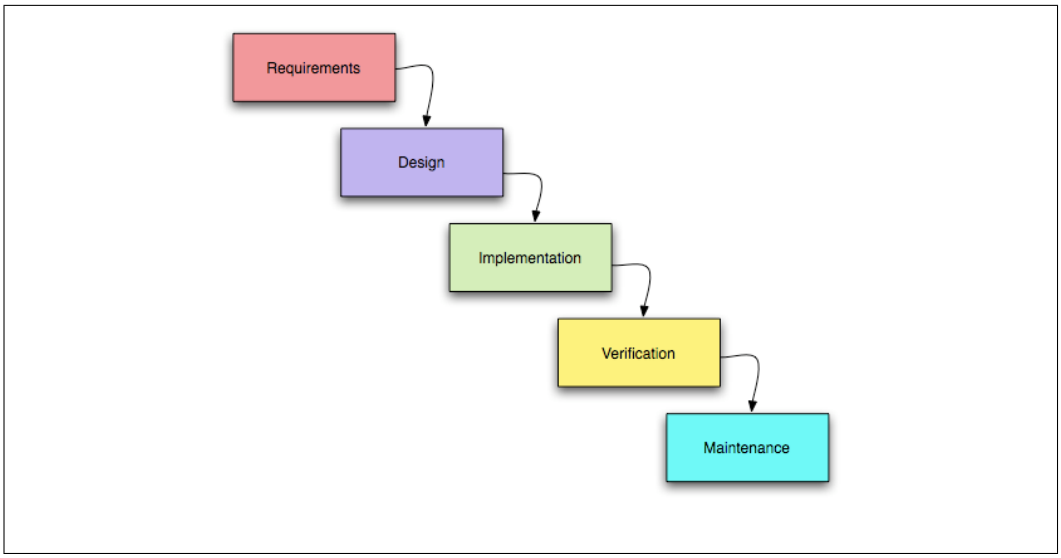


Figura 3.1: *Ciclo de vida en cascada[18]*

El matemático Barry W. Boehm[7] propuso a finales de los ochenta un ciclo de vida en espiral, Figura 3.2, compuesto por iteraciones. En cada iteración se construye una nueva versión del producto software. Los procesos de desarrollo que utiliza este ciclo de vida son los llamados procesos de desarrollo iterativo o por prototipos.

En la actualidad, y sobre todo en los proyectos de software que se basan en el paradigma de la orientación a objetos, se suelen dividir las aplicaciones en pequeños proyectos que se desarrollan en iteraciones. Se comienza implementando una parte reducida y sencilla de la aplicación, denominada prototipo, cuyos resultados permiten refinar progresiva e iterativamente las fases anteriores, corrigiendo errores y proponiendo mejoras, dando lugar así a nuevas versiones del producto cada vez más completas y complejas. Los beneficios de utilizar un proceso de desarrollo iterativo e incremental son:

- Disminuir el riesgo de obtener una aplicación inservible. Esto se debe a

que el prototipo resultante al final de cada iteración puede ser evaluado con lo que es posible decidir si se continúa o no.

- Se pueden realizar pruebas al final de cada iteración, sin necesidad de esperar a tener toda la aplicación desarrollada como ocurre en el ciclo de vida clásico.
- No es necesario que los requisitos estén completamente definidos al comienzo de una iteración.
- Se pueden desarrollar las principales funcionalidades del sistema, sin tener que tratar aspectos menos importantes, los cuales pueden realizarse en las últimas iteraciones.
- Se puede gestionar la complejidad de un modo sencillo, ya que no hay pasos largos ni complejos.
- El conocimiento adquirido en una iteración puede ser utilizado en iteraciones posteriores.

Las características del ciclo en cascada, hacen que sea una mala opción para este sistema. En primer lugar, la definición de requisitos en una aplicación web es una tarea complicada porque entra en juego aspectos de diseño que suelen ser ambiguos y difíciles de definir. Para añadir más dificultad, el sistema debe ser usable para usuarios con distintos perfiles informáticos. En segundo lugar, los sistemas que se desarrollan para terceras personas tienen que tener una retroalimentación constante por parte de los desarrolladores y el cliente. Si esto no ocurre, se tiene el riesgo de implementar algo totalmente inútil. Por lo tanto, se decidió utilizar un ciclo de vida en espiral.

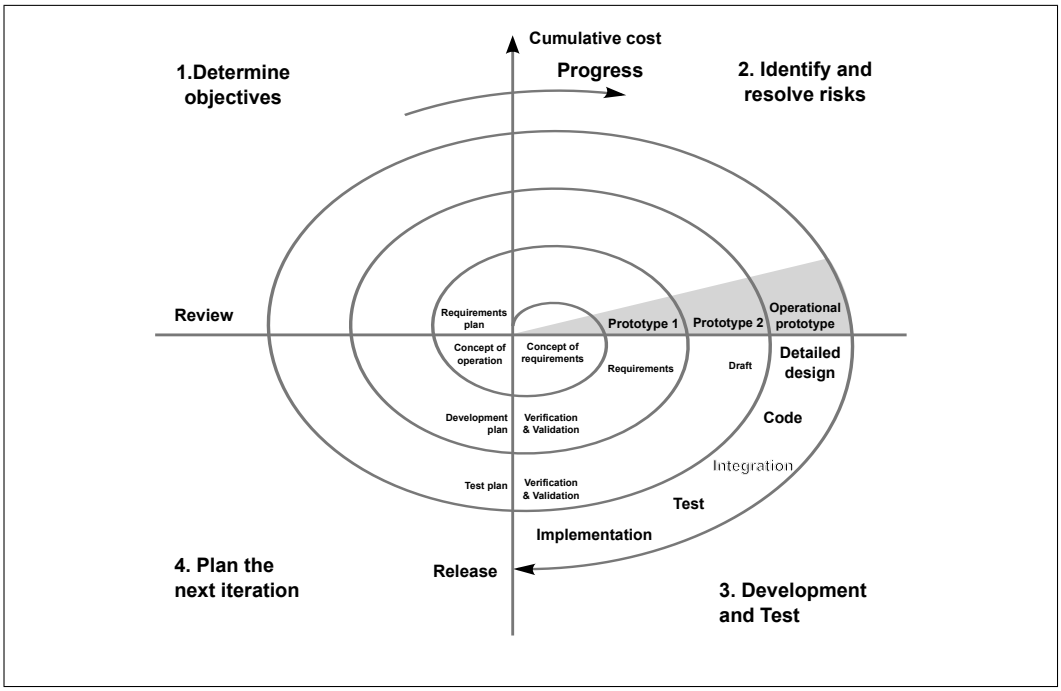


Figura 3.2: *Ciclo de vida en espiral[19]*

3.2. Fase de Análisis

En esta primera fase se debe realizar un análisis de requisitos. La finalidad es descubrir, refinar, modelar y especificar los requisitos que ha de satisfacer el sistema. Es decir, el análisis de requisitos tiene que identificar de forma precisa las funcionalidades que debe llevar a cabo la aplicación, sin tener en cuenta el modo en el que se van a implementar.

Para ello, en primer lugar, se definen los límites del sistema, esto es, lo que queda dentro y fuera del mismo. A continuación, se identifican los agentes externos o actores que van a interactuar con la aplicación. Y, por último, se especifican los servicios que ofrece el sistema a los usuarios. Estos requisitos se encuentran en el capítulo 3.3.

Para llevar a cabo la última tarea del análisis de requisitos se utilizan los casos de uso, son las historias de uso del sistema para satisfacer los objetivos de los usuarios, lo cual permite descubrir los requisitos del sistema de un modo más simple.

3.2.1. Identificación de los Actores

Los actores representan los agentes externos que interactúan con el sistema desarrollado. Un actor puede ser una persona, un sistema informático o una organización.

En el caso de esta aplicación existen tres tipos de actores. A continuación, se especifica cada uno de ellos:

- Anónimo. Representa aquella persona que introduce en su navegador web la URL de la aplicación, pero no se registra en ella. Este usuario podrá realizar una consulta y ver la ayuda de la aplicación.

- Lingüista. Representa aquella persona que accede al sistema, porque tiene un usuario con credenciales de lingüista. Este actor no tiene capacidad de realizar todas las operaciones del sistema. Pordrá realizar las siguientes operaciones: realizar consulta, insertar datos, modificar aquellos datos que previamente haya insertardo, ver la ayuda de la aplicación.

- Administrador. Representa aquella persona que accede al sistema con credenciales de Administrador. Este tipo de usuario puede realizar todas las operaciones ofrecidas por el sistema: realizar consulta, insertar datos, modificar datos, añadir usuario, eliminar usuario, realizar backup, cargar backup, ver la ayuda de la aplicación.

3.2.2. Casos de Uso

Como se indicó anteriormente, los casos de uso permiten entender y especificar los requisitos del sistema, es decir, permiten identificar todas las funciones u operaciones que el sistema deberá llevar a cabo para satisfacer las necesidades de los usuarios.

El diagrama de casos de uso, muestra los actores y el conjunto de operaciones que se han identificado para la aplicación. La Figura 3.3 representa el diagrama de casos de uso del proyecto utilizando la notación UML[10], *Unified Modeling Language*, donde las operaciones se representan mediante óvalos.

[10] `http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/`

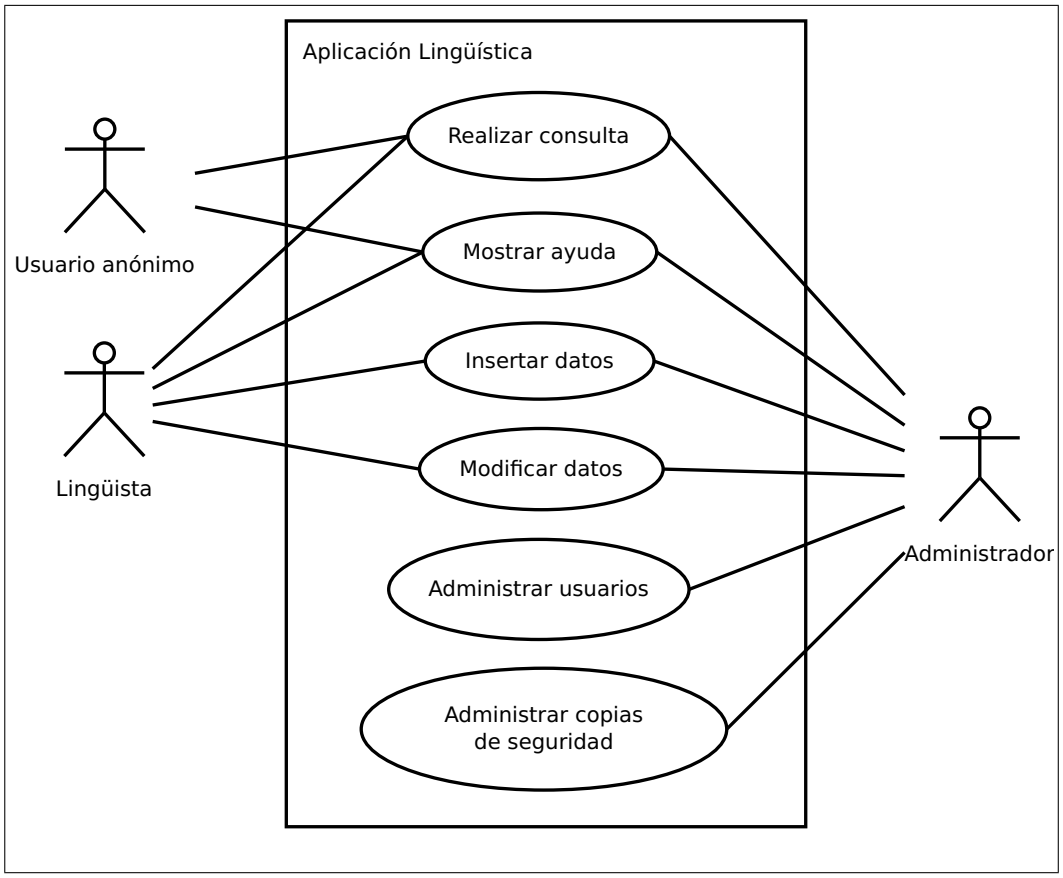


Figura 3.3: *Diagrama de casos de uso*

A continuación, se describen en formato breve cada uno de los casos de uso identificados:

- Realizar consulta. El usuario decide comenzar una consulta, para ello pulsa en el enlace correspondiente en el menú de la aplicación. Seguidamente, el sistema presenta el formulario de consulta. Una vez que el usuario haya rellenado el formulario, conforme a lo que desee consultar, este deberá pulsar el botón de «Buscar». Dependiendo de aquello rellenado por el usuario, el sistema presentara una tabla con la información asociada, o bien, un mensaje informando de que no existe información para la consulta pedida.

- Insertar datos. El usuario pulsa en el enlace «Insertar datos» del menú de la aplicación. Seguidamente, si el usuario tiene permisos, el sistema muestra un listado con todos los conceptos lingüísticos que la aplicación permite insertar. El usuario debe elegir uno de ellos. A continuación, el sistema muestra un formulario de inserción que el usuario debe de rellenar con aquello que crea conveniente. Una vez rellenado el formulario, el usuario debe pulsar en el botón Insertar. A continuación, el sistema muestra un mensaje informando si la inserción se ha realizado correctamente o no.

- Modificar datos. El usuario pulsa en el enlace «Modificar datos» del menú de la aplicación. Seguidamente, si el usuario tiene permisos, el sistema muestra un listado con todos los conceptos lingüísticos que la aplicación permite modificar. El usuario debe elegir uno de ellos. A continuación, el sistema muestra un formulario donde el usuario debe elegir aquel concepto específico que quiere modificar. Si el usuario elige un concepto que él no ha insertado, el sistema muestra un mensaje

informando que no tiene los derechos suficiente para modificar ese dato. En caso contrario, el sistema muestra un último formulario para realizar la modificación. El usuario debe pulsar en el botón «Modificar» para que la modificación se realice correctamente. A continuación, el sistema mostrará un mensaje informando si la modificación se ha realizado correctamente o no.

- Añadir usuario. El usuario con permisos de administrador pulsa en el enlace «Administración de usuarios» del menú de la aplicación. Después el sistema muestra tres enlaces: añadir un usuario, eliminar un usuario y mostrar usuarios. El usuario pulsa en el enlace de añadir usuario, y el sistema muestra un formulario para realizar la inserción del usuario. Una vez rellenado el formulario, el usuario pulsa el botón «Insertar». Después el sistema mostrará un mensaje informando si la operación se ha realizado correctamente o no.

- Eliminar usuario. El usuario con permisos de administrador pulsa en el enlace «Administración de usuarios» del menú de la aplicación. Seguidamente, el sistema muestra tres enlaces: añadir un usuario, eliminar un usuario y mostrar usuarios. El usuario pulsa en el enlace de eliminar usuario, y el sistema muestra un desplegable donde aparecen todos los usuario del sistema. El usuario selecciona aquel usuario que desea eliminar y pulsa el botón «Eliminar». Después el sistema muestra un mensaje informando si la operación se ha realizado correctamente o no.

- Mostrar usuarios. El usuario con permisos de administrador pulsa en el enlace «Administración de usuarios» del menú de la aplicación. Posteriormente, el sistema muestra tres enlaces: añadir un usuario, eliminar un usuario y mostrar usuarios. El usuario pulsa en el enlace de mos-

trar usuarios. El sistema muestra una tabla donde aparecen el id del usuario, nombre del usuario e email del mismo.

- Realizar backup. El usuario con permisos de administrador pulsa en el enlace «Administración de base de datos» del menú de la aplicación. Posteriormente, el sistema muestra dos enlaces: uno de ellos para realizar la copia de seguridad, y otro para cargar una copia de seguridad ya existente. El usuario pulsa en el enlace de realizar copia de seguridad. El sistema muestra un formulario. Una vez que el usuario rellena el formulario, pulsa en el botón «Realizar backup». Después el sistema muestra un mensaje informando si la copia de seguridad se ha realizado correctamente o no.

- Cargar backup. El usuario con permisos de administrador pulsa en el enlace «Administración de base de datos» del menú de la aplicación. Seguidamente, el sistema muestra dos enlaces: uno de ellos para realizar la copia de seguridad, y otro para cargar una copia de seguridad ya existente. El usuario pulsa en el enlace de cargar copia de seguridad. El sistema muestra un formulario. Una vez que el usuario rellena el formulario, pulsa en el botón «Realizar backup». Después el sistema muestra un mensaje informando si la operación se ha realizado con éxito o no.

- Mostrar ayuda de la aplicación. El usuario decide ver la ayuda de la aplicación, para ello pulsa en el enlace correspondiente en el menú de la aplicación. Posteriormente, el sistema muestra el manual de ayuda de la aplicación.

3.3. Especificación de Requisitos Software

3.3.1. Introducción

En este capítulo se recoge la Especificación de Requisitos Software (ERS) de la aplicación desarrollada en el presente proyecto. Esta sección se ha realizado siguiendo el estándar *IEEE Standard 830-1998*.

3.3.1.1. Propósito

El objetivo de este documento es el de definir de una manera clara y concisa las funcionalidades y restrincciones del sistema que se desea construir. Esta especificación se ha obtenido gracias a las diversas revisiones realizadas por el grupo de usuarios, hasta alcanzar su aprobación. Una vez aprobado servirá de base para realizar la construcción del sistema.

3.3.1.2. Ámbito del sistema

Se pretende desarrollar una aplicación web que permita a cualquier usuario consultar conceptos lingüísticos sobre palabras del idioma español. Además de la característica anterior, esta aplicación permite que lingüistas autorizados puedan insertar nuevos elementos en la base de datos de la aplicación.

3.3.1.3. Acrónimos y Definiciones

Se muestra una serie de definiciones, que servirán de ayuda para entender mejor el documento:

Acrónimo	Descripción
HTML	eXtensible Hypertext Markup Language, lenguaje extensible de marcado de hipertexto.
PDF	Portable document format, formato de documento portátil.
Web	World Wide Web.
GPL	GNU General Public License, licencia pública general de GNU.
HTTP	Hypertext Transfer Protocol, protocolo de transferencia de hipertexto.

Tabla 3.1: *Descripción de los acrónimos utilizados*

3.3.1.4. Visión general del documento

Este documento consta de tres secciones. Esta sección es la Introducción y proporciona una visión general del documento. En la segunda sección, se da una descripción general del sistema, con el fin de conocer las principales funciones que debe realizar, datos asociados, restricciones, supuestos y dependencias que afectan al desarrollo. En la tercera sección, se define con más detalle los requisitos que debe satisfacer el sistema.

3.3.2. Descripción general de la aplicación

En esta sección se presenta una descripción de alto nivel del sistema. Se presentarán las principales áreas de negocio, las funciones que el sistema debe

realizar, la información utilizada, las restricciones y otros factores que afecten al desarrollo del mismo.

3.3.2.1. Perspectiva del producto

El producto no se ha desarrollado completamente desde cero. Existía un diseño de la base de datos y su contenido léxico estaba almacenado empleando el gestor de base de datos Microsoft Access[11].

El producto, en principio, no interactuará con ningún otro sistema informático.

3.3.2.2. Funciones del sistema

En términos generales, el sistema deberá proporcionar las siguientes capacidades:

- Existirán tres tipos de usuarios: administrador, lingüista y usuario anónimo. Cada uno de ellos tendrá diferentes privilegios.
- Consulta de la información almacenada en la base de datos.
- Inserción de datos por parte de usuarios.
- Modificación de datos por parte de usuarios.
- Administración de los usuarios de la aplicación por parte de los usuarios administradores.

[11]Sitio web del gestor. http://office.microsoft.com/es-es/access/

- Administración de las copias de seguridad de la base de datos de la aplicación por parte de los usuarios administradores.

Por otro lado, los aspectos no funcionales del sistema serán los siguientes:

- El sistema será accesible vía web.
- Usabilidad de la aplicación. Se pretende que cualquier tipo de usuario con conocimientos mínimos de navegación web sea capaz de poder utilizar toda la funcionalidad del sistema.

A continuación se describen con más detalle estos aspectos.

Tres tipos de usuarios

El sistema deberá implementar el manejo de tres tipos de usuarios: administrador, lingüista y usuario anónimo. Cada tipo de usuario tendrá unos privilegios específicos. A continuación se especifican cada uno de estos privilegios:

- Usuario anónimo. Podrá realizar todo tipo de consultas, obteniendo toda la información que exista en la base de datos de la aplicación. Este tipo de usuario no podrá realizar: inserción o modificación de datos, administración de usuarios y administración de las copias de seguridad de la base de datos.
- Usuario lingüista. Este tipo de usuario, aparte de poder realizar todo tipo de consultas como el usuario anónimo, tiene la capacidad de realizar

inserciones sobre la base de datos de la aplicación. Sólo podrá modificar aquellos datos que propio usuario haya insertado en la aplicación. Este usuario no podrá administrar ni los usuarios de la aplicación, ni las copias de seguridad de la base de datos.

- Administrador. Este usuario tiene todos los privilegios posibles, es un usuario *superusuario* o *root*, y por tanto puede realizar todas las operaciones de las que dispone la aplicación. Estas operaciones son: realizar consultas, insertar datos, modificar datos, administración de usuarios y administración de las copias de seguridad de la base de datos.

Consulta de la información almacenada en la base de datos

La base de datos está formada por varias tablas relacionadas entre ellas. El objetivo es que a partir de uno o dos formularios tipo web, se pueda acceder a la información almacenada.

Toda la información consultada aparecerá en formato tabla HTML. Esta información tiene que poder ser exportable a formato PDF y Microsoft Office Excel.

Inserción de datos

Los usuarios de tipo lingüista y administrador tendrán la capacidad de insertar datos en la base de datos. La inserción de datos se realizará mediante formularios web. Por cada inserción de un dato, se guardará la fecha y el id del usuario que haya insertado ese dato.

Modificación de datos

Los usuarios de tipo lingüista sólo podrán modificar datos que hayan sido insertados por ellos mismos. Los usuarios de tipo administrador podrán modificar cualquier dato de la aplicación. Al igual que en la inserción de datos, la modificación de datos se realizará mediante formularios web. También se guardará la fecha y el identificador del usuario.

3.3.2.3. Características de los Usuarios

La mayor parte de los usuarios tendrás formación lingüista. Pero sus conocimientos técnicos pueden ser más o menos amplios. Por lo tanto, el sistema deberá ofrecer una interfaz de usuario estándar, fácil de aprender y sencilla de manejar. Lo deseable es que un usuario nuevo, con unas capacidades básicas en la navegación web, pueda familiarizarse con el sistema en menos de media hora.

3.3.2.4. Suposiciones y Dependencias

Suposiciones

En el documento se expresan los requisitos en términos de lo que el sistema debe proporcionar a los usuarios que acceden a él. No obstante, el sistema no podrá proporcionar ningún dato, si no existe ninguna base de datos que tenga dichos datos. Por tanto, los administradores se encargarán de proporcionar una base de datos relacional con los datos de la aplicación.

Dependencias

Este sistema no se comunica con ningún otro tipo de sistemas, por lo tanto no existen dependencias con otros sistemas.

3.3.3. Requisitos

En este apartado se presentan los requisitos funcionales, es decir: servicios, prestaciones, características, puntos de menú, etc. También se presentan los requisitos no funcionales: requisitos de calidad, fiabilidad, seguridad, usabilidad, etc. Estos dos tipos de requisitos, deberán ser cumplidos por el sistema. Cada uno de los requisitos tiene un nombre identificativo, además, los requisitos funcionales poseen un nombre breve.

Cada requisito tiene una prioridad de implementación que puede ser alta, media o baja. La prioridad alta, significa que la carencia de la misma no sería tolerable. La prioridad media es para aquellas que necesitan discusión. La priridad baja es para aquellas que no son necesarias ahora, pero que deberán incorporarse a medio-corto plazo.

3.3.3.1. Requisitos Funcionales

Consulta de la información

- **Req(01)consulta.recuperación**. **Prioridad Alta**.
  El sistema permitirá la recuperación del conjunto de datos, almacenados en la base de datos a partir de la elección de una serie de campos. La elección de estos campos se realizará mediante un formulario web

que el usuario deberá rellenar o seleccionar. El formulario de consulta tendrá la capacidad de impedir aquellas combinaciones que no tengan sentido dentro del ámbito lingüístico.

- **Req(02)consulta.presentación**. **Prioridad Alta**.
  El sistema presentará los datos en formato tabla HTML. En el encabezado de la tabla, aparacerán los campos que el usuario ha elegido.

- **Req(03)consulta.paginación**. **Prioridad Alta**.
  Cuando la consulta realizada por el usuario sobrepase el número de filas óptimo para que la navegación web sea fluida, se mostrará un formulario donde se podrá elegir el rango de filas que deben aparecer en la tabla HTML. Además, aparecerá un enlace para pasar a la siguiente página de la tabla.

- **Req(04)consulta.presentación de filtro**. **Prioridad Media**.
  En las opciones del formulario de consulta, que tengan datos en las cajas de texto pero no se haya pulsado en el check de validación, la consulta filtrará según los datos introducidos pero el campo no aparecerá en la tabla HTML.

- **Req(05)consulta.vacía**. **Prioridad Alta**.
  Cuando los datos insertados por el usuario no obtienen información alguna por parte de la aplicación, se mostrará un mensaje alertando de esta situación.

- **Req(06)consulta.ordenación**. **Prioridad Alta**.
  Todas las consultas que devuelvan un conjunto de datos, podrán ser ordenadas por el usuario pulsando en cualquiera de los encabezados de la tabla.

- **Req(07)consulta.exportación**. **Prioridad Media**.
  Todas las consultas que devuelvan un conjunto de datos, podrán ser exportados a los formatos: PDF, Microsoft Office Excel[12]. La exportación se realizará en la misma pantalla en la que aparece la tabla resultante. Para ello, aparecerán dos iconos, uno relacionado con la exportación a PDF y el otro relacionado con la exportación a formato Excel. Si se pulsa en uno de estos iconos se realizará la exportación.

Inserción de datos

- **Req(08)inserción.lista**. **Prioridad Alta**.
  En la primera pantalla de la sección de inserción de datos, aparecerá un mensaje informativo de esta sección.

- **Req(09)inserción.alta**. **Prioridad Alta**.
  El sistema permitirá la inserción de datos a los usuarios de tipo lingüista y administrador. La inserción de los datos se realizará mediante formularios web. En la base de datos se guardará el nombre del usuario que ha realizado la inserción.

- **Req(10)inserción.información**. **Prioridad Alta**.
  Se informará en otra pantalla si la inserción se ha realizado con éxito.

Modificación de datos

- **Req(11)modificación.lista**. **Prioridad Alta**.
  En la primera pantalla de la sección de modificación de datos, aparecerá

[12]Sitio web. http://office.microsoft.com/en-us/excel/

un mensaje informativo de esta sección.

- **Req(12)modificación.cambio**. **Prioridad Alta**.
  El sistema permitirá la modificación de datos a los usuarios de tipo administrador. Si el usuario es de tipo lingüista, se realizará una comprobación de que es el autor del dato a modificar. La modificación de los datos se realizará mediante formularios web.

- **Req(13)modificación.fallo autor**. **Prioridad Alta**.
  Cuando un usuario de tipo lingüista quiere realizar una modificación sobre un dato del que no es autor, aparecerá un mensaje alertando de esta situación e impedirá la modificación del dato.

- **Req(14)modificación.información**. **Prioridad Alta**.
  Se informará en otra pantalla si la modificación se ha realizado con éxito.

Administración de usuarios

- **Req(15)usuarios.privilegios**. **Prioridad Alta**.
  El sistema maneja tres tipos de usuarios: administrador, lingüista y usuario anónimo. A continuación, se describen las operaciones que puede realizar cada tipo usuario:

  - Usuario anónimo. Los usuarios anónimos tan sólo podrán realizar consultas a la base de datos de la aplicación.
  - Usuario lingüista. Este usuario podrá insertar datos en la base de datos de la aplicación. Además tendrá la capacidad de modificar aquellos datos que haya insertado. Al igual que el usuario anónimo, también podrá realizar consultas de datos.

- Administrador. Es el usuario con el máximo de privilegios, por tanto, podrá realizar todas las operaciones soportadas por el sistema. Tendrá la capacidad de realizar consultas, insertar datos, modificar datos, administrar usuarios y administrar las copias de seguridad de la aplicación.

- **Req(16)usuarios.ingreso**. **Prioridad Alta**.
  En todas las páginas web de la aplicación aparecerá un pequeño formulario, colocado en la esquina derecha de la pantalla, para poder autenticarse en la aplicación.

- **Req(17)usuarios.error en el ingreso**. **Prioridad Alta**.
  Cuando un usuario introduce un nombre de usuario o contraseña erróneo, aparecerá un mensaje informado del error.

- **Req(18)usuarios.alta**. **Prioridad Alta**.
  El alta de un nuevo usuario sólo puede ser realizado por un usuario de tipo administrador. El alta se realiza mediante un formulario en el que se deben rellenar los siguientes datos: nombre de usuario, correo electrónico, contraseña y tipo de usuario dentro del sistema.

- **Req(19)usuarios.información alta**. **Prioridad Alta**.
  Se informará en otra pantalla si el alta del nuevo usuario se ha realizado con éxito.

- **Req(20)usuarios.listar usuarios**. **Prioridad Alta**.
  En la sección de administración de usuarios, existirá un apartado donde la aplicación mostrará una tabla con los usuarios dados de alta en el sistema.

Administración de copias de seguridad

- **Req(21)backup.inicio**. **Prioridad Alta**.
  En la pantalla principal de la sección de Administración de copias de seguridad, aparecerá una pantalla informando de esta sección. Aparecerán dos enlaces, uno para realizar la copia de seguridad y otro para cargar una copia de seguridad existente.

- **Req(22)backup.realizar backup**. **Prioridad Alta**.
  La copia de seguridad se realizará rellenando un formulario simple. Existirán dos posibilidades: descargarse localmente la copia de seguridad, o bien, guardar la copia de seguridad en una carpeta del servidor. Esta carpeta no es fija, por lo tanto, tiene que ser especificada por el usuario.

- **Req(23)backup.información del backup**. **Prioridad Alta**.
  Se informará en otra pantalla si la copia de seguridad se ha realizado con éxito.

- **Req(24)backup.cargar**. **Prioridad Alta**.
  La carga de la copia de seguridad se realizará rellenando un formulario simple. El fichero de la copia de seguridad se puede cargar bien desde una carpeta del servidor donde se encuentre el backup o desde un archivo local. Tanto si el archivo backup se encuentra en el servidor o localmente en el ordenador, el usuario deberá especificar la ubicación correcta del mismo.

- **Req(25)backup.información de la carga**. **Prioridad Alta**.
  Se informará en otra pantalla si la carga de la copia de seguridad se ha realizado con éxito.

Ayuda

- **Req(26)Ayuda.inicio**. **Prioridad Alta**.
  La aplicación tendrá siempre visible una sección de ayuda. Dicha sección, servirá para explicar sobre todas las funcionalidades de la aplicación.

- **Req(27)Ayuda.contextual**. **Prioridad Alta**.
  La aplicación tendrá una ayuda contextual siempre visible, que explicará las características principales de la página donde se encuentre el usuario.

3.3.3.2. Requisitos de Interfaces Externos

Interfaces de Usuario

- **Req(28)**. **Prioridad Alta**.
  La interfaz de usuario debe ser orientada a formularios accesibles a través de un navegador web.

Interfaces Hardware

- **Req(29)**. **Prioridad Alta**. El sistema deberá funcionar sobre diferentes plataformas hardware (Intel[13], AMD[14], IBM PowerPC[15], etc).

[13]Sitio web. http://intel.com/
[14]Sitio web. http://amd.com/
[15]Sitio web. http://power.org/

Interfaces Software

- **Req(30). Prioridad Alta.**
  La aplicación será multiplataforma, por tanto deberá funcionar sobre sistemas de tipo GNU/Linux[16] y Windows XP[17] en adelante.

- **Req(31). Prioridad Alta.**
  Toda la tecnología y estándares utilizado para la realización del sistema debe tener licencia GPL o equivalente.

- **Req(32). Prioridad Alta.**
  El servidor deberá permitir al lenguaje web utilizado realizar copia de seguridad de la base de datos.

- **Req(33). Prioridad Alta.**
  La aplicación deberá poder acceder al sistema de archivos para guardar y cargar las copias de seguridad de la base de datos.

Interfaces de Comunicación

- **Req(34). Prioridad Alta.**
  Se usará la red e infraestructura de la Facultad de Informática de la Universidad Politécnica de Madrid[18].

Requisitos de Rendimiento

- **Req(35). Prioridad Alta.**

[16]Sitio web del sistema operativo. http://linux.org/
[17]Sitio web del sistema operativo. http://www.microsoft.com/spain/windowsxp/default.mspx
[18]Sitio web. http://www.fi.upm.es/

El tiempo de respuesta será la esperada en cualquier aplicación de tipo web. Es decir, la respuesta de la aplicación a cada petición web no podrá superar los 3 ó 4 segundos.

Requisitos de Desarrollo

- **Req(36). Prioridad Alta.**
  El ciclo de vida elegido para el desarrollo del sistema, es el de prototipo evolutivo. De esta forma, se podrá incorporar de manera más sencilla cambios y nuevas funcionalidades en el sistema.

Requisitos Tecnológicos

- **Req(37). Prioridad Alta.**
  Se requiere un servidor web con un Sistema de Gestión de Bases de Datos y un servidor web HTTP.

3.3.3.3. Atributos

Integridad de la información

- **Req(38). Prioridad Alta.**
  EL sistema guardará una copia de seguridad del estado de la base de datos. Mediante esta copia, se podrá restaurar el sistema en su totalidad.

Mantenibilidad

- **Req(39). Prioridad Alta.**
  El sistema debe requerir el mínimo posible de mantenimiento. Todas las funciones posibles de mantenimiento se deberán poder realizar vía web.

Seguridad

- **Req(40). Prioridad Alta.**
  Se podrá acceder al sistema mediante una validación de usuario y contraseña.
- **Req(41). Prioridad Media.**
  Todos los accesos de usuarios serán registrados en un fichero de log. Este fichero mostrará la fecha y ahora de acceso al sistema.

3.4. Fase de Diseño

3.4.1. Diseño de la Base de datos

La aplicación a desarrollar necesita un sistema de gestión de datos que permita almacenar, recuperar, actualizar y eliminar la información de los elementos que pueden ser persistentes en el sistema.

Para el diseño del modelo de datos se creó en primer lugar un diagrama entidad-relación. Dado que se optó por seguir un modelo relacional para la base de datos, este diagrama después se transformó en un conjunto de tablas que correspondiera al esquema de la base de datos. Estos elementos se describen en los subapartados que restan.

3.4.1.1. Diagrama entidad-relación

En un primer momento, existía un diagrama entidad-relación[9]. Pero debido a las modificaciones usuales debido al proceso de refinamiento, sufrió una serie de modificaciones. En la Figura 3.4 se muestra el diagrama entidad-relación final. Los rectángulos representan las entidades, los círculos atributos de entidades o relaciones, los rombos las relaciones entre entidades y los números, la cardinalidad de las relaciones, es decir, el número de ocurrencias de las entidades con las que puede estar relacionada la ocurrencia de una entidad. En los siguientes párrafos se describen los elementos más importantes.

- Unidad léxica. Se define unidad léxica como forma lingüística o con-

junto de formas lingüísticas que posee significado propio. Una unidad léxica puede ser base de alguna función léxica con uno o varios lemas como valores. De la misma manera, puede ser base de alguna función léxica con uno o varias etiquetas semánticas. Además una unidad léxica puede pertenecer a uno o varios conceptos y puede ser actante de uno o varios lemas.

- Lema. Representa cada una de las entradas de un diccionario, puede tener uno o varios rasgos, semantemas, esquemas o locuciones. Puede ser actante de uno o varias unidades léxicas y corresponder a uno o varios conceptos. Por último, puede pertencer a una o varias etiquetas semánticas.

- Concepto. Esta entidad almacena un sistema de categorías o conceptos de un dominio, en concreto, el mundo en el que vivimos. Es una entidad que tiene una relación reflexiva en la que un concepto puede pertenecer a uno o varios conceptos. De esta forma, se obtiene una tabla que sirve para clasificar los conceptos del dominio. Un concepto puede formar parte de ninguna o varias etiquetas semánticas, unidades léxicas o lemas. Y de igual manera le ocurre a una etiqueta semántica, unidad léxica o lema.

- Etiqueta semántica. Una etiqueta semántica identifica, generalmente, el inmediato genérico que se emplea en su definición. Tiene una relación reflexiva en la que una etiqueta semántica puede pertenecer a una o varias etiquetas semánticas, de esta manera, se obtiene una clasificación de etiquetas semánticas. Además una etiqueta semántica se relaciona con uno o varios conceptos. Una etiqueta semántica puede suponer una o varias funciones léxicas.

- Función léxica. Una función léxica pone en relación dos unidades léxicas, una de ellas recibe el nombre de base, y la otra se llama valor. Por ejemplo, la función léxica Real1 se define como «utilizar algo de acuerdo con su destino». Así, esta función léxica relacionaría la base herramienta con el valor usar o juguete con jugar. Los valores son glosas de esa función léxica, cuando no son específicos de una base, sino de un conjunto de ellas, que están clasificadas en una misma etiqueta semántica.

- Paráfrasis. Algunas se corresponden con funciones léxicas que admiten alguna explicación en lenguaje natural. Cada una de las cuales constituiría una paráfrasis de la función léxica. Por ejemplo, Real podría aceptar las paráfrasis «hacer lo que se espera que haga» y «hacer que algo cumpla su función o finalidad».

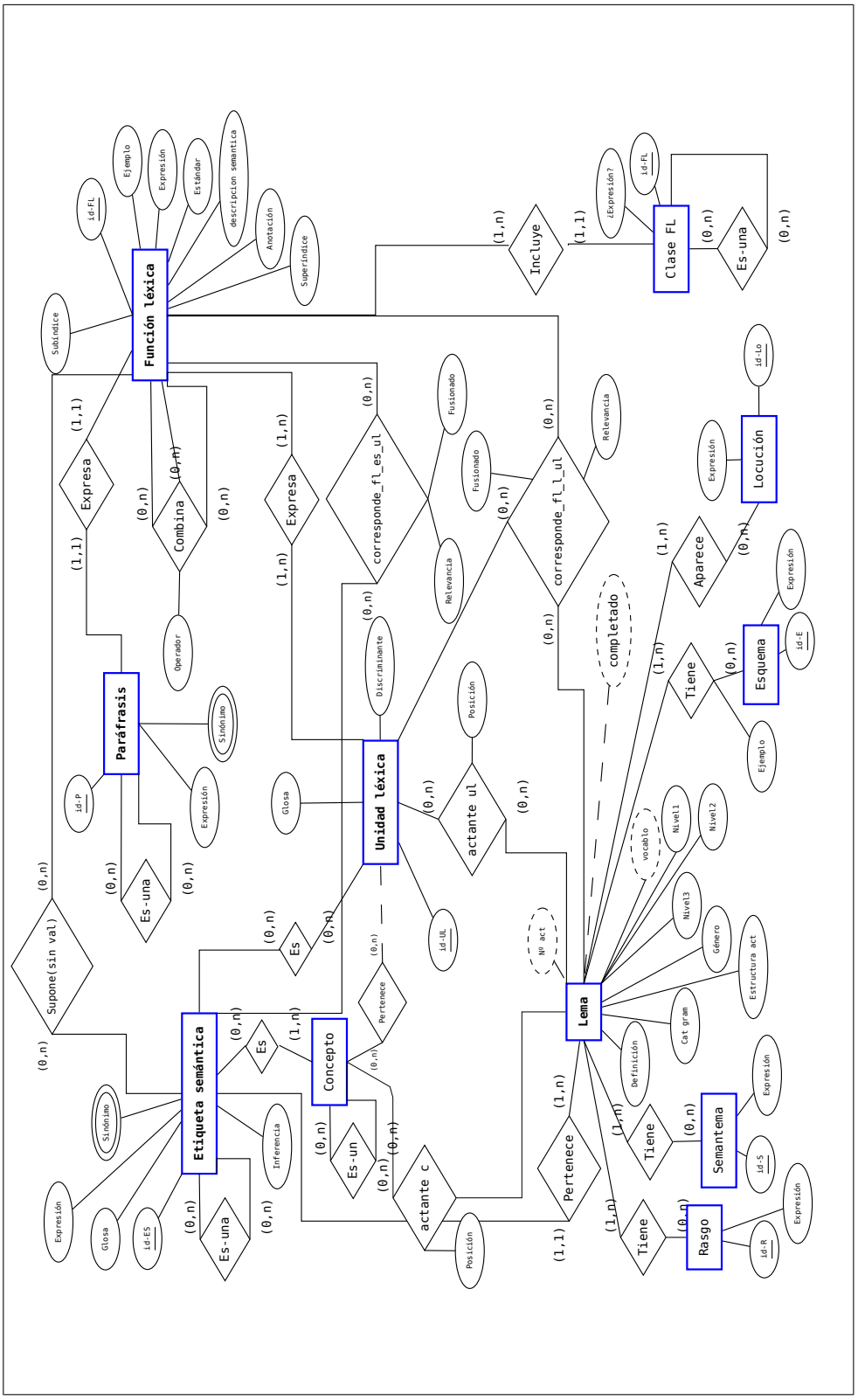


Figura 3.4: *Diagrama entidad-relación final*

La entidad «Usuario» no se ha incorporado en la Figura 3.4 para no enmarañarla, pero sí existe en el modelo. Un usuario puede alterar una o varias entidades del modelo (se entiende por «alterar» a insertar o modificar datos). Por este motivo, la entidad «Usuario» se relaciona con todas las entidades del modelo, véase Figura 3.5.

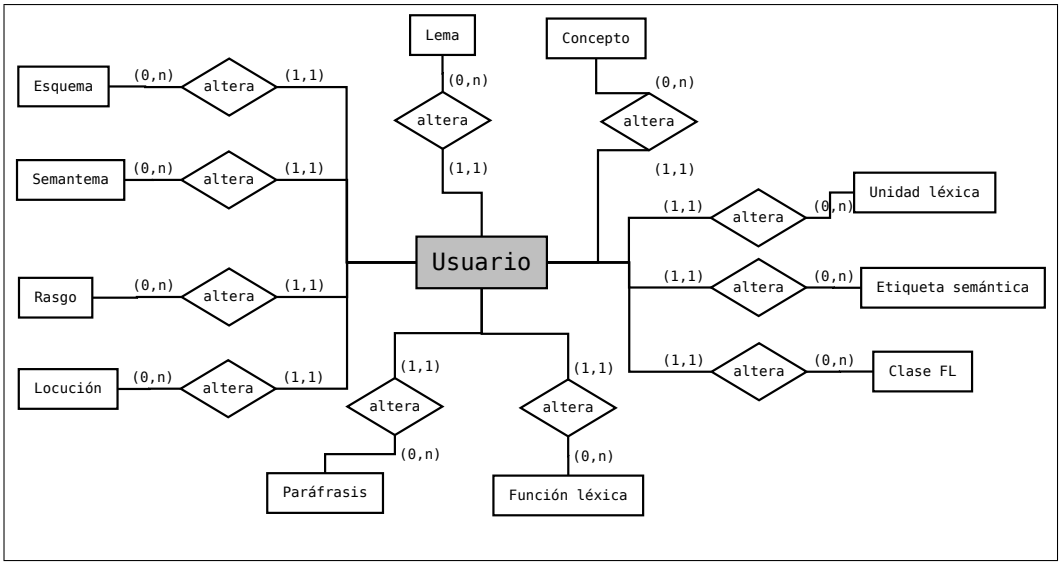


Figura 3.5: *Relaciones de la entidad «Usuario» con el resto de entidades del modelo*

3.4.1.2. Paso a tablas

Una vez terminado el diagrama entidad-relación, es necesario realizar lo que se conoce como paso a tablas[8]. En este proceso cada entidad del diagrama se corresponde con una tabla. Además de las entidades, las relaciones de tipo M:N generan una tabla intermedia que tiene como clave primaria la unión de los identificadores de las entidades relacionadas. En la Figura 3.6, se han coloreado en azul las tablas intermedias, obteniendo 31 tablas finales. Las líneas punteadas indica que la relación existente es de tipo 1:1, las líneas

punteadas y lisas se corresponden con relaciones 1:N, mientras que las líneas completamente lísas son relaciones M:N. En la Figura 3.6, no se ha incluido la tabla «Usuarios» para no enmarañar el diagrama, ya que todas las tablas tienen una clave foránea que apunta a la clave primaria de la tabla «Usuarios». De esta manera, se puede identificar qué usuario ha modificado cierta tabla del modelo.

El fin de este proceso, es que el gestor de la base de datos sea capaz de relacionar la información tal y como se ha diseñado en el diagrama entidad-relación.

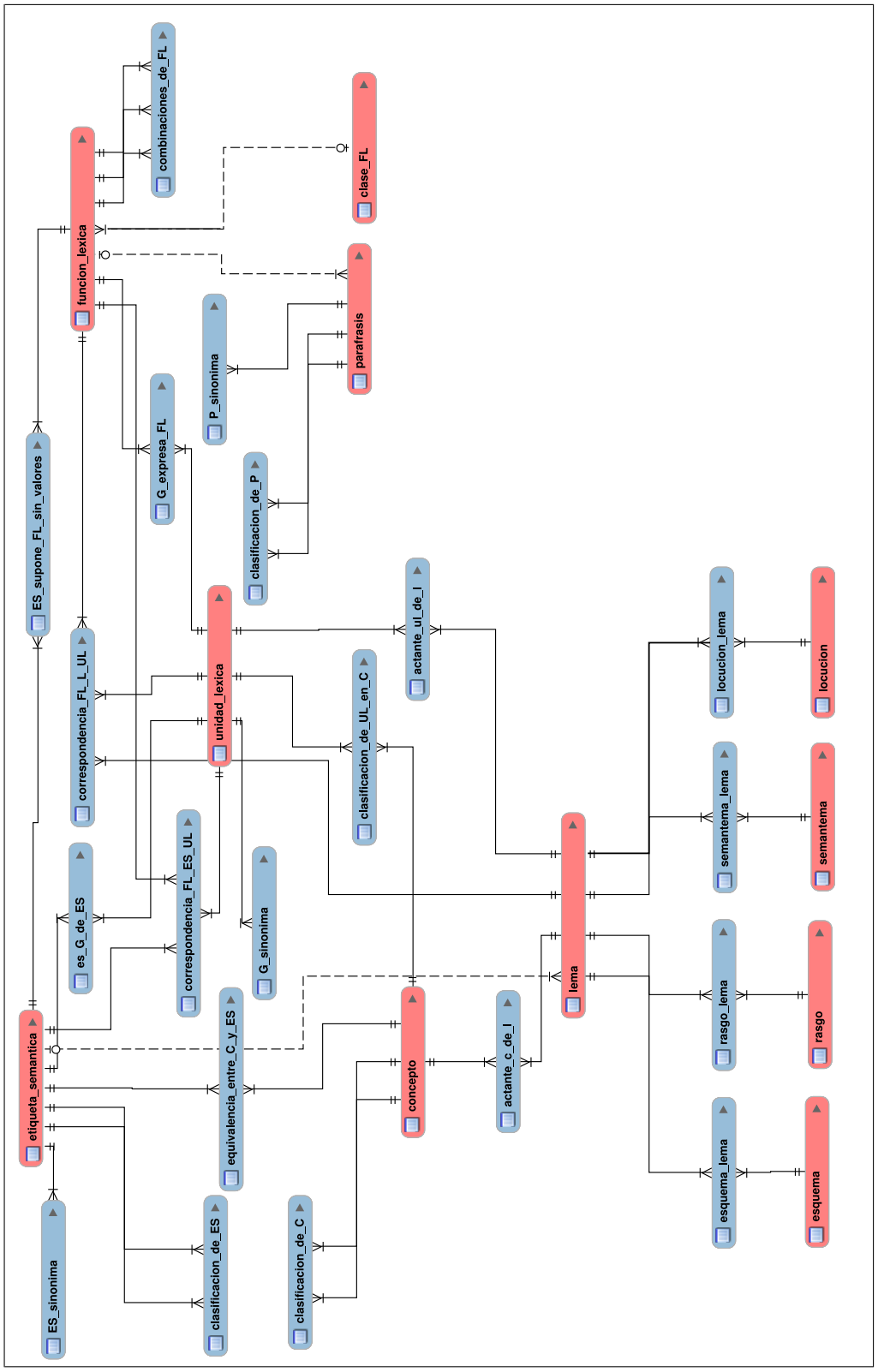


Figura 3.6: *Tablas de la base de datos*

3.4.1.3. Gestor de la Base de datos

Existen varios sistemas de gestión de bases de datos relacionales. Los más utilizados dentro del software libre son: MySQL[19] y PostgreSQL[20], ambos son multihilo y multiusuario. Los puntos que predominaron para la elección de MySQL fueron los siguientes:

- Es una aplicación en la que predomina las consultas únicamente de acceso, en donde no se modifican los datos de las mismas. En este tipo de consultas, MySQL ofrece un mejor rendimiento que PostgreSQL.
- MySQL consume menos recursos que PostgreSQL, aunque es menos estable cuando se observa una alta tasa de accesos. A priori los accesos esperados en el sistema no llegarán a una tasa tan alta.
- Otro punto a favor, fue el conocimiento del gestor y de las herramientas que tiene para administrar correctamente la base de datos.

A partir de los puntos mencionados anteriormente, se puede deducir que en la decisión tomada, se premió la alta velocidad de MySQL en consultas con poca carga de trabajo. Por tanto, se elegió MySQL como el sistema gestor de la base de datos.

[19]Sitio web. http://www.mysql.com
[20]Sitio web. http://www.postgresql.org

3.4.2. Módulos del Sistema

La creación de un módulo está estrechamente ligada a la estructuración del código fuente, para que este sea más legible y productivo. Por definición, los módulos agrupan un conjunto de subprogramas y estructuras de datos que conceptualmente tienen cierta homogeneidad.

Para la toma de decisión de los módulos del sistema, fue de gran ayuda analizar el diagrama de casos de uso que se realizó en la fase de Análisis del Sistema. Tras este análisis se crearon los siguientes módulos, Figura 3.7:

- Consultas. Este módulo agrupa el funcionamiento de la sección de consultas. Fundamentalmente, su función es analizar el formulario rellenado por el usuario, obtener y preparar los datos para que se muestren correctamente en la tabla de salida.
- Download. Este módulo actúa en aquellos casos en los que se desea descargar un fichero generado por la aplicación. Por ejemplo, cuando se desea descargar una copia de seguridad de la base de datos.
- Dump. Se ocupa de la sección de administración de la base de datos del sistema. Realiza y carga copias de seguridad de la base de datos.
- Export. Este módulo genera archivos de tipo Microsoft Excel[21] o PDF a partir de unos datos recibidos. Se utiliza cuando se quiere exportar la salida obtenida a partir de una consulta.
- Index. Implementa el acceso de diferentes roles de usuario en el sistema.

[21] Sitio web. http://office.microsoft.com/en-us/excel/

- Insertar. Agrupa el funcionamiento de la sección de inserción de datos de la aplicación. Se encarga de analizar todos los formularios de inserción e insertar los datos introducidos por el usuario en la base de datos del sistema.

- Modificar. Se encarga del funcionamiento de la sección de modificación de datos de la aplicación. Analiza todos los formularios de modificación, comprueba que el usuario conectado a la aplicación tenga permisos suficientes para modificar el dato y modifica los datos de la base de datos del sistema.

- Usuarios. La función de este módulo es administrar los usuarios del sistema, implementando las operaciones de crear y eliminar usuarios.

El objetivo principal de los módulos «Export» y «Download», es que sean utilizados por otros módulos. El primero de ellos, lo utiliza el módulo «Consultas» para exportar los resultados de una consulta a formatos tipo Microsoft Excel y PDF. Por otra parte, el módulo «Download» se encarga de implentar la funcionalidad de poder descargar un fichero que se encuentra en el servidor del sistema. Aunque sólo es utilizado por el módulo «Dump» para descargar la copia de seguridad de la base de datos, se pensó que podía ser utilizado en otro momento. Por este motivo se desarrolló como un módulo aparte.

En Symfony, los controladores de todos los módulos creados, se generalizan a una única clase llamada *sfActions*, que se encarga de todos los eventos web. Debido a esto, todos los módulos dibujados en la Figura 3.7 se enlazan con una flecha a *sfActions*. Cuando un módulo hace uso de otro módulo, se representa mediante una línea discontinua. Por ejemplo, el módulo «Dump» hace uso del módulo «Download».

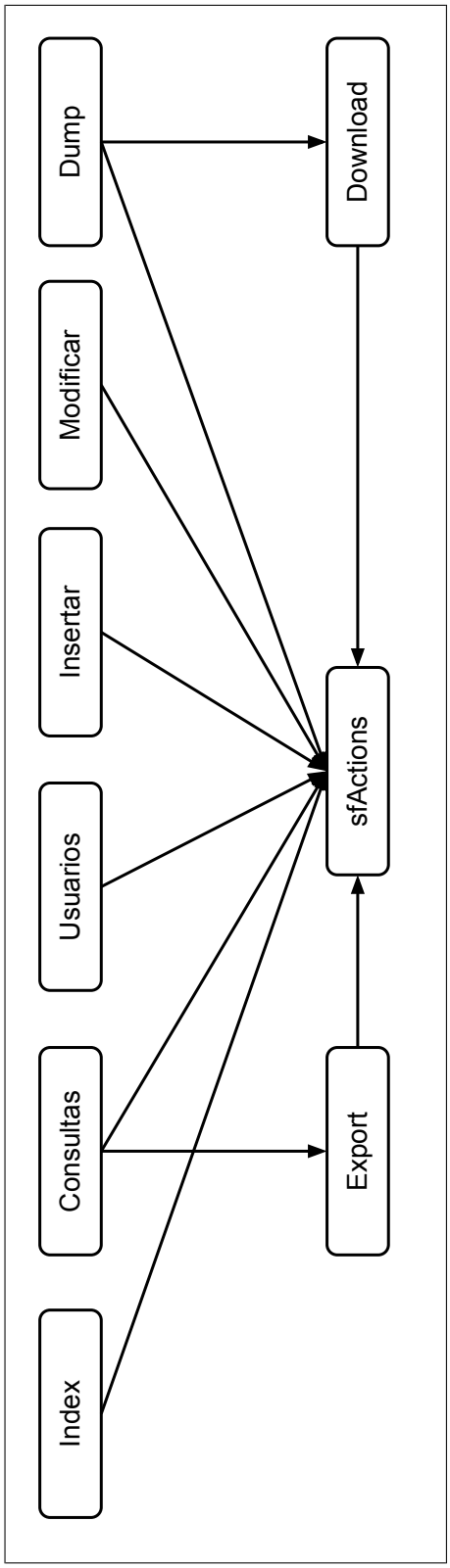


Figura 3.7: *Módulos del Sistema*

3.4.3. Interfaz web

Antes de comenzar con el diseño de la aplicación, se revisó el documento de Especificación de Requisitos Software, capítulo 3.3, para analizar qué necesidades se correspondían con aspectos de la interfaz de la aplicación. Resumiendo, se hace referencia a las siguientes exigencias:

- El acceso de aplicación se realiza en línea mediante un navegador web.
- La interfaz de la aplicación debe estar dirigida a todo tipo de usuarios.
- El tiempo de carga entre páginas no debe superar los 4 segundos.
- Tiene que existir un enlace siempre visible que permita acceder a la sección de ayuda.
- Es necesario una sección de acceso de usuarios.

A la hora de realizar la interfaz de cualquier aplicación lo más complicado es realizar un diseño intuitivo, accesible y usable, ya que son aspectos muy poco tangibles. Para ayudar en este aspecto existen guías y normas con los criterios generales que ayudan a determinar lo que habrá que tener en cuenta en esta interfaz, sin olvidar, obviamente, lo recogido en la ERS.

La interfaz de la aplicación se ha diseñado siguiendo una única plantilla, véase Figura 3.8, utilizando tres colores básicos. De esta manera, la homogeneidad y uniformidad del diseño facilita la ubicación de la información de cara al usuario.

Las partes fundamentales de la plantilla son:

1. Acceso de usuarios. Formulario para el acceso de los usuarios registrados.
2. Identidad. Título de la aplicación.
3. Barra de navegación. Aparecen todas las secciones en que se divide la aplicación.
4. Título. Representa el título de la sección donde se encuentra el usuario.
5. Contenido de la sección.
6. Píe de página.

Por desgracia la interpretación del código HTML y CSS de los navegadores web no es idéntica, sobre todo en los navegadores *Microsoft Internet Explorer con respecto al resto*[22], se ha intentado que las diferencias visuales y funcionales sean mínimas. Por este motivo, se ha querido generar código HTML y CSS lo más puro posible, para que todas las versiones de los navegadores que se utilizan actualmente, puedan acceder a la aplicación de forma correcta.

[22] `http://windows.microsoft.com/es-ES/internet-explorer/products/ie/home`

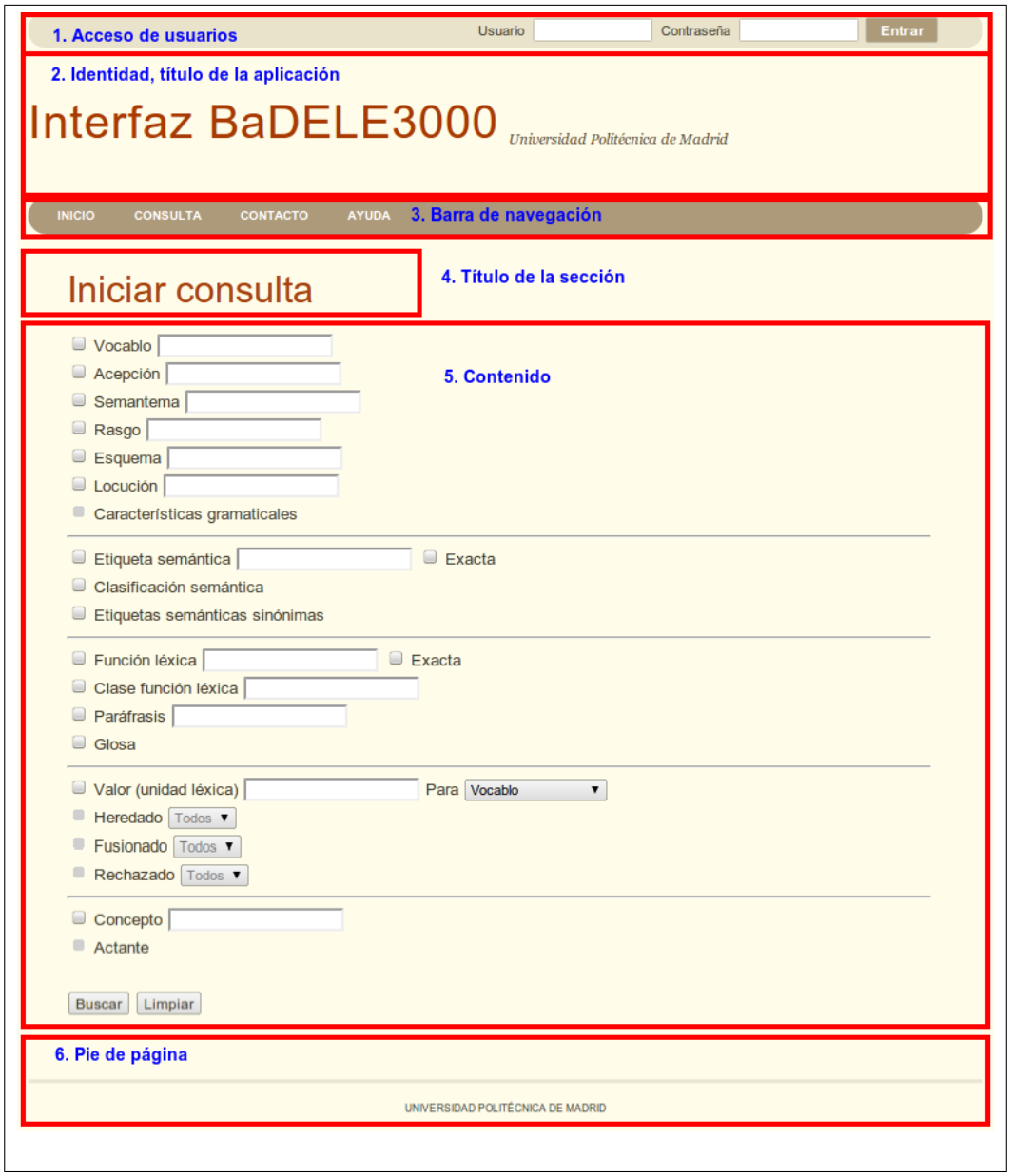


Figura 3.8: *Estructura del diseño web de la aplicación*

3.5. Fase de Implementación

En la primera sección de este capítulo, se especifican las tecnologías utilizadas para la implementación del sistema. Posteriormente, se describen las tareas llevadas a cabo y decisiones tomadas.

3.5.1. Tecnologías, lenguajes y estándares empleados

Esta sección trata de las tecnologías, lenguajes y estándares empleados en la creación de la aplicación.

3.5.1.1. HTML

Este lenguaje se empleó para crear la aplicación web del sistema.

HTML, siglas de *HyperText Markup Language* es un lenguaje de marcado creado en 1989 por Tim Berners-Lee, y es el predominante para la realización de páginas web. Se denominan lenguajes de marcado, a aquellos lenguajes, que entre el texto de un documento, usan una serie de etiquetas para incorporar una información adicional sobre la estructura o presentación del mismo.

A través de estas etiquetas, los navegadores web saben cómo presentar la información del documento conforme a la estructura ideada por el autor. Estas etiquetas se identifican porque están encerradas mediante los símbolos <>. La versión actual del lenguaje es HTML 5 y el desarrollo de la misma está regulado por el Consorcio W3C, *Worl Wide Web Cosortium.*

3.5.1.2. CSS

CSS[23], siglas de *Cascading Style Sheets*, en español Hojas de Estilo en Cascada, es un lenguaje utilizado para especificar la apariencia y el formato de un documento escrito en un lenguaje de marcado. El objetivo principal de este lenguaje es separar el contenido de la presentación del mismo. Para ello, se encarga de definir propiedades de fuentes, color, fondo, texto y clasificación de las etiquetas del lenguaje de marcado utilizado. Actualmente el lenguaje se encuentra en la tercera versión, lanzada en el año 2010.

Una de las grandes ventajas de utilizar CSS, es la posibilidad de que las personas con visibilidad disminuida puedan aumentar el tamaño de letra de la página web que están visualizando. Este hecho se debe a que los navegadores, permiten que los usuarios modifiquen su propia hoja de estilo local que será aplicada a cierto sitio web. Otras de las ventajas, es que se puede definir un tipo de hoja de estilo diferente según el dispositivo o el usuario que la muestre. De este modo, se puede utilizar diferentes hojas de estilo para dispositivos móviles o para sintetizadores de voz, ofreciendo una gran versatilidad.

Este lenguaje se utilizó dentro del sistema para la creación de la aplicación web. Se incorporó junto con HTML, en un único archivo para dar formato, estructura y aparencia más estética a la aplicación web. En la Figura 3.9, se muestra la estructura final de cada de las páginas web que engloba la aplicación. Para más información, véase la sección 3.4.3.

[23]Sitio web. http://www.w3.org/Style/CSS/

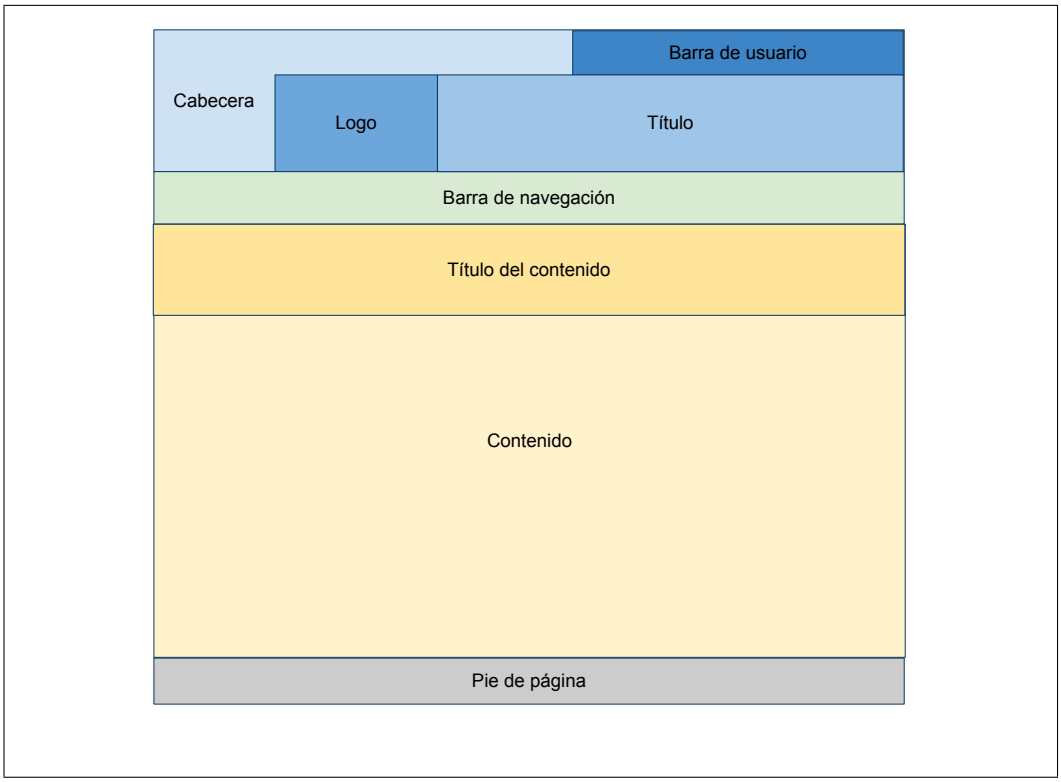


Figura 3.9: *Estructura de la página web*

3.5.1.3. PHP

PHP[24] proviene de un programa anterior, llamado PHP/FI. Fue creado por Rasmus Lerdorf en 1995, inicialmente como un conjunto de archivos de órdenes en lenguaje Perl[25]. Rasmus decidió liberar el código fuente de PHP/FI para que cualquiera pudiese utilizarlo, así como arreglar errores y mejorar el código.

Existen varias ventajas que acreditan a este lenguaje como uno de los mejores para realizar páginas web modernas. Una de estas primeras ventajas

[24]Sitio web. http://php.net
[25]Sitio web del lenguaje Perl. http://www.perl.org/

es que es multiplataforma y por tanto puede funcionar en sistemas operativos UNIX[26] (y similares como Linux[27], Mac OS X[28]) y Windows[29]. También es capaz de entenderse con los servidores webs y gestores de bases de datos más comunes. Otra de las ventajas, es que tiene una API (*Application Programming Interface*) o interfaz de progración con gran funcionalidad y bien documentada. Además tiene la característica de expandir su funcionalidad utilizando módulos.

En cuento al paradigma de programación, PHP no obliga a utilizar ningún tipo concreto. De hecho, es posible utilizar programación estructurada u orientada a objetos. Esta aplicación se ha desarrollado utilizando la programación orientada a objetos, además de usar el Modelo Vista Controlador mediante el *framework* Symfony, que se explicará más tarde en este mismo capítulo.

3.5.1.4. jQuery

jQuery[30] es una biblioteca de JavaScript[31] de código abierto, diseñado para simplificar la utilización de JavaScript desde el lado del cliente HTML. Su primer lanzamiento fue a principios de 2006 y su creador es John Resig. Se estima que el 41 % de sitios web utilizan tecnología jQuery, siendo por tanto la biblioteca más popular.

La finalidad de jQuery es facilitar la navegación y aumentar la interacción

[26]Sitio web del sistema operativo. http://www.unix.org/
[27]Sitio web del sistema operativo. http://www.linux.org/
[28]Sitio web del sistema operativo. http://www.apple.com/es/macosx/
[29]http://windows.microsoft.com
[30]Sitio web del *framework*. http://jquery.com
[31]Sitio web del lenguaje. https://developer.mozilla.org/en/JavaScript

entre usuario y página web. Para ello, jQuery maneja una serie de efectos que proporcionan un aspecto más dinámico al sitio web. A continuación se comentan algunos de estos efectos:

- Eliminación de contenido web, encerrado entre marcas div.
- Control de los atributos de elementos HTML que tengan un id concreto. jQuery es capaz de modificar los atributos de estos de elementos de manera sencilla.
- Creación de menús desplegables
- Validación de formularios

Dentro del ámbito del sistema, se ha utilizado jQuery en la mayoría de formularios de la aplicación web. Sobre todo en las secciones de inserción y modificación de datos.

3.5.1.5. Apache

Apache[32] es un servidor web HTTP de código abierto. Es capaz de funcionar en plataformas basadas en UNIX[33] así como en Windows[34]. Es un servidor muy modular, separado por un núcleo y varios módulos que añaden funcionalidad al servidor haciéndolo mucho más versátil.

Para la realización de la aplicación web se utilizó este servidor web por ser software libre. La utilización de otro tipo de servidor web no debe influir en el funcionamiento de la aplicación.

[32]Sitio web. http://www.apache.org
[33]Sitio web del sistema operativo. http://www.unix.org/
[34]Sitio web del sistema operativo. http://windows.microsoft.com/

3.5.1.6. MySQL

Existen varios sistemas de gestión de bases de datos relacionales. Los más utilizados dentro del software libre son: MySQL[35] y PostgreSQL[36], ambos son multihilo y multiusuario. Los puntos que predominaron para la elección de MySQL fueron los siguientes:

- Es una aplicación en la que predomina las consultas únicamente de acceso, en donde no se modifican los datos de las mismas. En este tipo de consultas, MySQL ofrece un mejor rendimiento que PostgreSQL.
- MySQL consume menos recursos que PostgreSQL, aunque es menos estable cuando se observa una alta tasa de accesos. A priori los accesos esperados en el sistema no llegarán a una tasa tan alta.
- Otro punto a favor, fue el conocimiento del gestor y de las herramientas que tiene para administrar correctamente la base de datos.

A partir de los puntos mencionados anteriormente, se puede deducir que en la decisión tomada, se premió la alta velocidad de MySQL en consultas con poca carga de trabajo. Por tanto, se elegió MySQL como el sistema gestor de la base de datos.

3.5.1.7. Patrón Modelo-Vista-Controlador

El Modelo-Vista-Controlador[37], a partir de ahora MVC, es un estilo de arquitectura software, utilizado actualmente como un patrón de diseño soft-

[35]Sitio web. http://www.mysql.com
[36]Sitio web. http://www.postgresql.org
[37]Steve Burbeck, How to use Model-View-Controller

ware que separa la lógica de la aplicación de la interfaz de usuario, lo que permite el desarrollo independiente de pruebas y mantenimiento de cada uno. El patrón se divide en tres estratos:

- Modelo. Maneja el comportamiento y los datos de la aplicación. Responde las solicitudes de información dependiendo del estado, normalmente del que tenga el controlador. En los sistemas que se basan en eventos, el modelo notifica cuándo cambia la información para que puedan reaccionar a dicho evento.
- Vista. Se ocupa de generar la imagen del modelo en un formato adecuado para la interacción con el mismo. Normalmente elementos de la interfaz de usuario. Pueden existir varias vistas de un modelo único para diferentes propósitos.
- Controlador. Recibe la entrada, generalmente de acciones de usuarios, e inicia una respuesta que, por lo general, son peticiones al modelo y en algunos casos a la vista.

En el caso de la aplicación, el controlador recibe una serie de GET/POST de entrada, a partir de la interacción del usuario con la interfaz. El controlador accede al modelo, si la acción lo requiere modificará el modelo, y delegará en la vista para generar la interfaz del usuario. Por tanto, la vista accede a los datos del modelo para desplegar la interfaz. En la Figura 3.10[38] se incluye la estructura del patrón:

[38]http://en.wikipedia.org/wiki/Model-view-controller

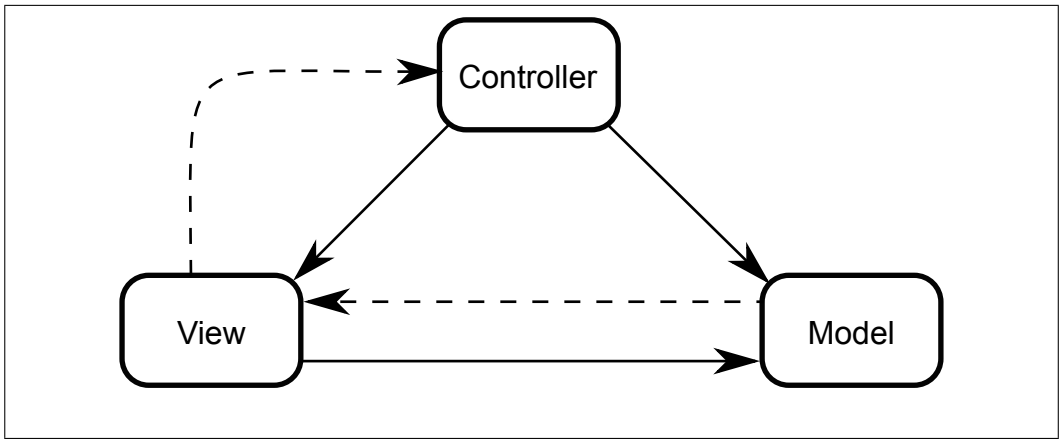


Figura 3.10: *Ilustración del patrón Modelo-Vista-Controlador*

Ciclo de vida del Modelo-Vista-Controlador

El primer paso en el ciclo de vida empieza cuando el usuario hace una solicitud al controlador con información sobre lo que el usuario desea realizar. Entonces el controlador decide a quién debe delegar la tarea y es aquí donde el modelo empieza su trabajo. En esta etapa, el modelo se encarga de realizar operaciones sobre la información que maneja para cumplir con lo que le solicita el controlador. Una vez que termina su labor, la información resultante de sus operaciones se transmiten al controlador, el cuál a su vez la redirige a la vista. La vista se encarga de transformar los datos en información visualmente entendible para el usuario. Finalmente, la representación gráfica es transmitida de regreso al controlador y éste se encarga de mostrársela al usuario. El ciclo entero puede empezar nuevamente si el usuario así lo requiere.

Ventajas y desventajas de MVC

Las principales ventajas de hacer uso del patrón MVC son:

- La separación entre el Modelo y la Vista, es decir, distinguir los datos de la representación visual de los mismos.
- Es mucho más sencillo agregar representaciones de los mismos datos o información.
- Facilita la creación de nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento de las otras capas.
- Facilita el mantenimiento en caso de errores.
- Facilita el proceso de pruebas de funcionamiento del sistema.

Las desventajas de seguir el planteamiento de MVC son:

- La separación de conceptos en capas agrega complejidad al sistema.
- La cantidad de archivos que hay que mantener y desarrollar se incrementa considerablemente.
- La curva de aprendizaje del patrón de diseño es más alta que usando otros modelos más sencillos.

La comparación de ventajas y desventajas de MVC puede ser un tema muy subjetivo y se puede prestar a debate como se puede mostrar en la cantidad

de resultados que ofrece Google[39] al buscar términos como: *is MVC the best solution?*, *is MVC the future*, *MVC vs MVA*.

3.5.1.8. Symfony

Symfony es un *framework* de aplicaciones web que sigue el Modelo-Vista-Controlador. Su primera publicación fue en el año 2005, bajo licencia MIT License, y por tanto se considera software libre. Está desarrollado en PHP 5, y por tanto, al igual que ocurre con PHP, Symfony es capaz de funcionar en plataformas basadas en UNIX (Linux, Mac OS X), así como en Windows. Es compatible con la mayoría de gestores de bases de datos como Oracle[40], MySQL, PosgreSQL, Microsoft SQL Server[41]. Actualmente existen varios sitios desarrollados con Symfony. Por ejemplo, caben destacar: Delicious[42], Yahoo Bookmarks[43], Dailymotion[44]. Aunque la aplicación se ha desarrollado con la versión 1.4 de Symfony, la última versión del *framework* es la 2. No se quiso desarrollar la aplicación en esta última versión porque existen bastantes diferencias entre la última versión y la penúltima (versión 1.4) del *framework*. Además como esta última versión se lanzó a medidados de 2011, la documentación no es muy amplia.

[39] http://google.com
[40] htttp://www.oracle.es
[41] http://www.microsoft.com/sqlserver/en/us/default.aspx
[42] http://delicious.com
[43] es.bookmarks.yahoo.com
[44] http://www.dailymotion.com

Capa de persistencia

Las aplicaciones que se desarrollan hoy en día estructuran y guardan la información en sistemas de gestión de datos con el objetivo de que puedan ser reutilizados, bien desde la misma aplicación o desde otro proceso diferente. La capa de persistencia es la pieza que permite almacenar, recuperar, actualizar y eliminar el estado de los elementos que pueden ser persistentes en uno o más sistemas gestores de datos.

El sistema gestor de datos puede corresponder a un modelo relacional, uno orientado a objetos o cualquier otro sistema, de manera que el programador no tenga que hacer ninguna traducción de los objetos para convertirlos en persistentes, sino que sea esta capa la que se encargue de esta transformación.

El uso de un modelo relacional en una aplicación que sigue el paradigma orientado a objetos, como ocurre en este proyecto, hace que sea necesario un componente de software que permita la comunicación de estos dos instrumentos.

La capa de persistencia traduce entre los dos modelos de datos: de registros (modelo relacional) a objetos (modelo orientado a objetos) y viceversa. Cuando el programa quiere grabar un objeto llama al motor de persistencia, que traduce el objeto a registros y llama a la base de datos para que guarde estos registros. De la misma manera, cuando el programa quiere recuperar un objeto, la base de datos recupera los registros correspondientes, los cuales son traducidos en formato de objeto por el motor de persistencia. De esta manera, el programa sólo ve que puede guardar objetos y recuperar objetos, como si estuviera programado para una base de datos orientada a objetos. La base de datos sólo ve que guarda registros y recupera registros, como si

el programa estuviera dirigiéndose a ella de forma relacional.

A esta última técnica de programación se le llama ORM (*Object-Relational Mapping*[45]). Un ORM es una capa que permite relacionar objetos con un modelo de datos relacional, oculta todo el mecanismo de conexión al gestor de la base de datos y además no tiene que escribir las sentencias SQL necesarias para hacer consultas y/o modificaciones a los registros de la base de datos. La Figura 3.11 recoge un ejemplo de programación ORM, donde una función realiza dos consultas a una base de datos, utilizando dos métodos que el propio objeto tiene definido:

```
public function getNombreCompleto()
{
  return $this->getNombre().' '.$this->getApellidos();
}
```

Figura 3.11: *Ejemplo de programación ORM*

Los métodos de acceso ocultan la lógica de los datos favoreciendo una programación más sencilla.

Symfony permite elegir entre dos ORMs diferentes: Doctrine y Propel. En la aplicación, se ha elegido Doctrine ya que es el configurado por defecto en Symfony 1.4.

[45]http://www.hibernate.org/about/orm

Modelo-Vista-Controlador en Symfony

Como se ha comentado anteriormente, Symfony está basado en el patrón de diseño Modelo-Vista-Controlador. La arquitectura del patrón separa la lógica de negocio (modelo) y la presentación (vista). Por tanto, el principio de la arquitectuta Modelo-Vista-Controlador es separar el código en tres capas, de acuerdo con su naturaleza. El código que trata la lógica de datos, se ubica dentro del modelo, el código de presentación dentro de la vista, y por último, la lógica de la aplicación dentro del controlador.

Symfony define como Modelo, la elección del ORM que puede ser: Propel o Doctrine como se ha comentado anteriormente. Dentro de la vista agrupa los Helpers y la clase sfView. Por último, el Controlador engloba las clases: sfController, sfFilter, sfAction. En la Figura 3.12, se muestra un esquema del patrón Modelo-Vista-Controlador en Symfony. Todas las clases y variables del núcleo de Symfony llevan el prefijo sf.

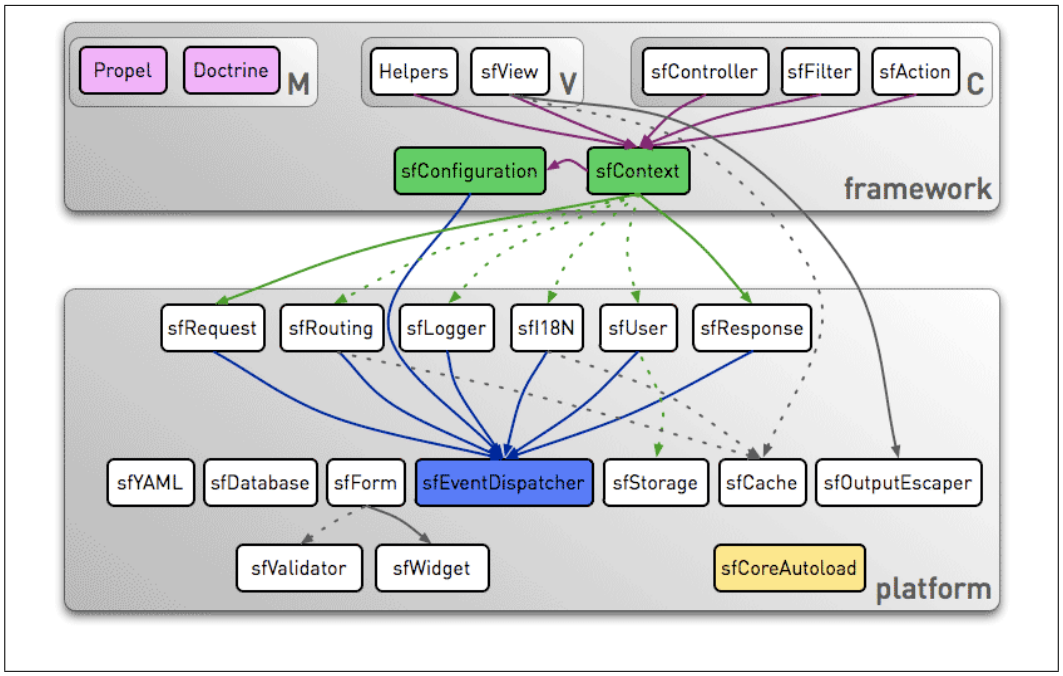


Figura 3.12: *Ilustración del patrón Modelo-Vista-Controlador en Symfony[11]*

Núcleo de las clases de Symfony

La implementación del patrón usa varias clases, entre las que destacan las recogidas en la Tabla 3.2.

Clase	Descripción
sfController	Es la clase controlador. Codifica la petición y la manda a la acción.
sfRequest	Es la clase que almacena todos los elementos de las peticiones. Como por ejemplo: parámetros, cookies, cabeceras.
sfResponse	Contiene las cabeceras de respuesta y contenido. Es el objeto que convertirá los datos a una respuesta HTML que devolverá al usuario.
sfView	Esta clase es aquella que controla la capa de la vista del modelo.
sfContext	El contexto (recuperado por sfContext:: getInstance ()) almacena una referencia a todos los los objetos principales y la configuración actual. Es accesible desde cualquier lugar.

Tabla 3.2: *Descripción de las principales clases de Symfony*

Todas las clases y variables del núcleo de Symfony llevan el prefijo **sf**.

Extensiones o *plugins* de Symfony

Symfony permite instalar extensiones o plugins, de esta manera se puede aumentar la funcionalidad del *framework*. En la página oficial del producto existe un apartado de extensiones que la comunidad va creando.

Para la realización de la aplicación se incluyó la extensión que crea un sistema para el control y acceso de usuarios. También se instaló una extensión para la creación de documentación, que se utilizó coon el fin de generar la exportación de las consultas en ficheros de tipo Microsoft Office Excel y PDF. También es posible instalar cualquier librería PHP de manera sencilla.

3.5.1.9. Subversion

Para la realizar el control de la configuración del sistema se empleó Subversion o simplemente SVN[46]. Un sistema de control de versiones, con licencia software libre, que automatiza los procesos de guardar, recuperar, identificar y mezclar versiones de archivos.

En los últimos años, Subversion, está gozando de más popularidad que otros sistemas de control de versiones más antiguos como por ejemplo CVS[47], *Concurrent Versions System*. El origen de este crecimiento de popularidad posiblemente se deba a que Subversion a diferencia de CVS, no tiene un identificador de revisión por archivo sino que tiene un número de revisión común para todo el repositorio. Otra de las diferencias más significativas con otros sistemas de control de versiones, es que Subversion no envía al servidor los ficheros completos sino que envía únicamente las diferencias que haya podido haber de una revisión del repositorio a otra, con la disminución de trabajo por parte del servidor que esto supone.

El sistema de control de versiones Subversion, se suele instalar en un servidor. En este servidor podrán existir varios repositorios o entornos donde

[46]http://subversion.tigris.org/
[47]http://www.cvshome.org/

se quiere realizar un control de versiones. Cada repositorio tendrá una dirección URL inequívoca. Por otro lado, cada usuario que quiera hacer uso del repositorio deberá instalarse un cliente Subversion en su máquina local.

En el caso concreto de este proyecto, se ha utilizado el servicio gratuito de Subversion ofrecido por la página `http://unfuddle.com/`.

3.5.2. Decisiones sobre diseño

Algunos detalles del diseño del sistema no se pudieron establecer sin tener en cuenta la fase de implementación. Así a lo largo de la codificación del sistema, se tuvieron que tomar ciertas decisiones cuyo impacto en el tiempo de la elaboración de la aplicación es sustancial.

Una de las decisiones más relevantes, se encuentra en la elaboración del módulo consultas. Este módulo tiene una complicación a nivel de codificación lógica bastante fuerte. Esta complicación, viene determinada por la cantidad de opciones que se puedan dar en el formulario principal de la sección de consultas. Este formulario, tiene veinte cajas de validación y cuatro cajas de selección con sus diferentes opciones.

Ante la situación expuesta anteriormente, la primera decisión que se tomó fue realizar un estudio de las consultas que lingüísticamente tenían sentido. Tras él, se analizó qué opciones y combinaciones del formulario debería seleccionar un usuario para obtener dichas consultas válidas. Una vez hecho esto, se tomó la decisión de que había que facilitar la interacción del formulario de consulta con el usuario, para que no se puediesen elegir consultas no deseadas. Después de este primer paso, se logró limitar bastante la cantidad

de posibles consultas que un usuario podía realizar.

Se plantearon dos posibilidades para programar las consultas permitidas:

- Concatenación de sentencias SQL[48]. Las opciones que el usuario ha pulsado en el formulario, se van analizando y generando pequeñas sentencias SQL que se van concatenando hasta formar una consulta SQL ejecutable en la base de datos del sistema.
- Una sentencia de control. Como todas las consultas que un usuario puede ejecutar mediante el formulario han sido analizadas, se puede nombrar cada consulta con un código inequívoco. Mediante una sentencia de control tipo *if-then-else*, es posible acceder a la consulta concreta, elegida por el usuario y generar la sentencia SQL que al final será ejecutada por la base de datos.

De las dos opciones explicadas anteriormente se escogió la primera de ellas, «Concatenación de sentencias SQL». La elección de esta alternativa se basó en que era más elegante y que a priori se presumía que fuese más rápida de programar ya que la cantidad de líneas de código sería menor. Aunque al principio parecía resultar una opción válida, conforme se iba programando y el número de concatenaciones crecía, resultaba más difícil el mantenimiento y un cambio requería bastante tiempo. Por lo tanto, se decidió cambiar la manera de codificar esta parte del sistema por la segunda opción antes mencionada, ya que era más fácil de mantener.

[48]Sitio web. http://www.sql.org

3.5.3. Órganización del código

Dado que se empleó Symfony, es necesario tener en cuenta cómo organiza el código generado. En un primer momento puede ser algo tediosa, ya que está dividida en varios directorios, sin embargo, resulta necesaria y muy útil. En este capítulo se explica en profundidad toda la estructura que Symfony utiliza.

En la mayoría de las aplicaciones web, existen ficheros comunes. Por ejemplo:

- Una base de datos
- Clases y librerías PHP
- Librerías externas desarrolladas por terceros
- Archivos estáticos (HTML, imágenes, JavaScript, hojas de estilos, etc.)
- Archivos que se ejecutan por lotes (batch files) que normalmente se ejecutan a través de la línea de comandos.
- Archivos de configuración
- Archivos de log

Para organizar todo este contenido, Symfony proporciona una estructura en forma de árbol, véase Tabla 3.3 que además tiene consistencia con la arquitectura modelo-vista-controlador mediante la agrupación proyecto/aplicación/módulo. Cada vez que se crea un proyecto, esta estructura es generada automáticamente.

Directorio	**Descripción**
apps/	En esta carpeta se encuentran todas las aplicaciones del proyecto.
cache/	Se encuentran los archivos de cache.
config/	Archivos de configuración del proyecto.
data/	En este directorio se almacenan los archivos relacionados con los datos. Por ejemplo, el esquema de una base de datos, el archivo que contiene las instrucciones SQL para crear la estructura de tablas, etc.
doc/	Carpeta donde se guarda documentación sobre el proyecto.
lib/	Las bibliotecas y clases del proyecto.
log/	Guarda todos los archivos de registro generados por Symfony. Symfony genera un archivo de log por cada aplicación y por cada entorno de trabajo.
plugins/	En esta carpeta se encuentran todos los plugins instalados en el sistema.
test/	Los archivos de pruebas unitarias y funcionales.
web/	El directorio raíz web. Los únicos archivos accesibles desde Internet están en este directorio.

Tabla 3.3: *Descripción de las carpetas del producto*

3.5.3.1. Estructura de la carpeta aplicación

En esta sección se profundiza más detalladamente en la carpeta de la aplicación, ya que es una de las más importantes:

- config. Almacena la mayor parte de la configuración de la aplicación, salvo aquella que sea general para el framework.
- i18n. Contiene todos los archivos necesarios para la internacionalización de la aplicación. Si la internacionalización se realiza mediante el uso de una base de datos con equivalencia entre lenguas, no se utilizará.
- lib. Contiene las clases y librerías utilizadas sólo por la aplicación.
- modules. Se almacenan los módulos de la aplicación.

Los métodos y atributos de las clases de una aplicación no están accesibles para otras aplicaciones.

3.5.3.2. Estructura de la carpeta módulo

Cada aplicación contiene uno o más módulos. Cada módulo tiene sus propios subdirectorios, véase Figura 3.13.

```
apps/
  [nombre de la aplicación]/
    modules/
      [nombre del módulo]/
        actions/
          actions.class.php
        config/
        lib/
        templates/
          indexSuccess.php
```

Figura 3.13: *Estructura de la carpeta módulo*

- actions/ . Suele contener un único archivo actions.class.php, que contiene la clase que agrupa todas las acciones del módulo.
- config/ . Contiene ficheros de configuración con los parámetros locales del módulo.
- lib/ . Almacena las clases y librerías específicas del módulo.
- template/ . Recoge las plantillas correspondientes a las acciones del módulo que como se ha mencionado anteriormente están en el archivo actions.class.php. Cuando se crea un módulo, Symfony genera de manera automática la plantilla indexSuccess.php. En este caso index se corresponde con la acción index y success es un convenio utilizado por Symfony. Todas las plantillas tendrán siempre este patrón de nombrado, nombre_acciónSuccess.php.

3.5.3.3. Estructura de la carpeta web

Es una carpeta con archivos de acceso público. Existen muy pocas restrincciones para organizar el contenido del directorio web, tan sólo habría que guardar cierta nomenclatura en los archivos que contiene la carpeta, para que todo sea más legible, véase Figura 3.14.

```
web/
  css/
  images/
  js/
  uploads/
```

Figura 3.14: *Estructura de la carpeta web*

Contiene las siguientes subcarpetas:

- css/. Contiene las hojas de estilos.
- images/ . Contiene las imágenes de tipo png, jpeg o gif, utilizadas para la creación de la aplicación web.
- js/ . Contiene los JavaScript de la aplicación.
- uploads/ . Contiene los archivos subidos por los usuarios.

Aunque la estructura planteada por Symfony no es la única posible, se ha considerado adecuada para el proyecto porque separa correctamente los diferentes tipos de ficheros de una aplicación web, característica muy recomendable de cara al futuro mantenimiento y modificación de la aplicación.

3.6. Fase de Pruebas

En esta sección se explican las pruebas que se han realizado para identificar fallos de implementación, calidad o usabilidad. De esta manera, se ha podido evaluar la calidad del producto desarrollado.

3.6.1. Pruebas Unitarias

Se consideran pruebas unitarias, aquellas que conciernen al funcionamiento de procedimientos o funciones utilizadas por cierto módulo.

El *framework* Symfony, permite la creación de pruebas unitarias de manera sencilla. Estas pruebas, son ficheros PHP que se pueden ejecutar individualmente o todos a la vez. Para un ejemplo sencillo de prueba unitaria, véase la Figura 3.15:

```
// La siguiente línea es común a todas las pruebas unitarias
require_once dirname(__FILE__).'/../bootstrap/unit.php';

$t = new lime_test(3);
$t->is(myConsultas::getVocablo('perro'), 'perro');
$t->is(myConsultas::getVocablo('alfa'), '');
$array_tmp = ['id_c' => 3, 'id_l' =>10];
$t->ok(myInserciones::insertarActanteCdeL($array_temp));
```

Figura 3.15: *Detalle de la creación de una prueba unitaria*

En el ejemplo de prueba unitaria, se comprueba que la función *getVocablo*

utilizado por el módulo «consultas», devuelve la cadena «perro» cuando se pasa el argumento «perro» y que devuelve vacío cuando se pasa el argumento «alfa». También se comprueba que la función *insertarActanteCdeL* del módulo *inserciones*, realiza la inserción sin errores, es decir, devuelve verdadero. Como en esta última prueba no se comparan valores, se utiliza el método «ok». Symfony tiene varios métodos[49] predefinidos para utilizar en las pruebas unitarias.

Estas pruebas unitarias se pueden ejecutar mediante la línea de comandos de la Figura 3.16. Los resultados se muestran por pantalla.

```
php symfony test:unit
```

Figura 3.16: *Ejemplo de cómo ejecutar una prueba unitaria*

3.6.2. Pruebas de caja negra

Las pruebas de caja negra se centran en los resultados que devuelve un módulo para una serie de entradas específicas. La parte más complicada en la realización de estas pruebas, es diseñar un amplio abanico de entradas que englobe aquellas situaciones en las que el módulo puede fallar. Por lo tanto, el diseñador debe elaborar entradas extremas que sean inesperadas para el módulo.

Mediante la realización de pruebas de caja negra, se comprobaron si los módulos comentados en la sección *Módulos del Sistema* devolvían los resultados deseados para ciertas entradas. A continuación, se explica el análisis realizado para elaborar este tipo de pruebas por cada sección de la aplicación.

[49]Métodos predefinidos. http://www.symfony-project.org/jobeet/1_4/Doctrine/es/08

Además, se ha realizado una tabla explicativa con las entradas, salidas esperadas y salidas obtenidas por cada prueba de cada sección, véase la Figura 3.30.

3.6.3. Realizar consulta

El núcleo de esta sección es un formulario web con unos 25 campos, en donde el usuario puede elegir múltiples opciones. Realizar la batería de pruebas de esta sección ha sido una tarea realmente complicada, ya que un formulario tan amplio permite una cantidad muy elevada de combinaciones.

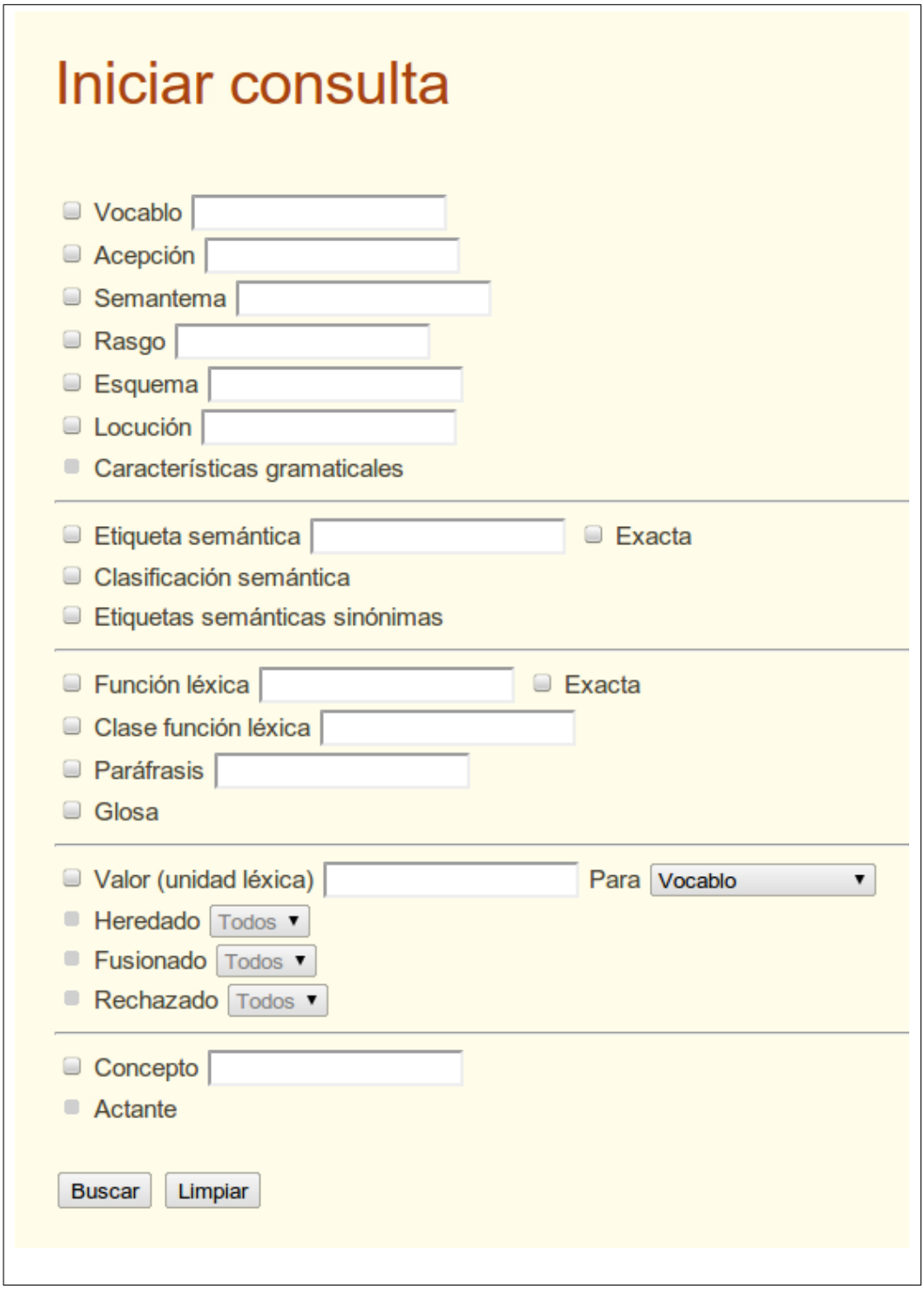


Figura 3.17: *Formulario de la sección: Realizar consulta*

Para poder acotar este número de combinaciones, se tomó la decisión de priorizar la realización de pruebas sobre las combinaciones que a priori, iban a ser las más solicitadas por los usuarios. También se realizaron pruebas de resistencia para verificar que la aplicación se comportaba correctamente ante situaciones atípicas.

Estas combinaciones más usuales, estaban íntimamente ligadas con las entidades más nucleares del diagrama entidad-relación descrito en la sección 3.4. Estas entidades son:

- Concepto
- Función léxica
- Etiqueta semántica
- Lema
- Unidad léxica

Sólo con la elección de estas entidades en el formulario anteriormente citado, se obtiene 32 combinaciones posibles que se tienen que comprobar. Si a estas combinaciones, se añaden opciones propias de las entidades citadas anteriormente, como por ejemplo en el caso de etiqueta semántica: heredado, fusionado, rechazado; el número de combinaciones aumenta aún más. Por este motivo, la batería de pruebas en esta sección es bastante densa.

A modo de ejemplo se incluye una de las pruebas llevadas a cabo.

3.6.3.1. Prueba realizar consulta

La prueba que a continuación se va a explicar, consta de una consulta en la que se selecciona varias opciones del formulario principal de la sección. En concreto, se buscarán los vocablos y funciones léxicas de la etiqueta semántica «Accesorio». A continuación, se detallan todos los pasos necesarios para obtener dicha información:

Paso 1

Estando en la página principal de la aplicación, se pulsa en «Iniciar consulta» del menú. La aplicación muestra un formulario con varias opciones. Para obtener los datos citados, se hace clic en las casillas de verificación: vocablo, etiqueta semántica y función léxica. Como además, se quiere filtrar por la etiqueta semántica «Accesorio», se inserta en la caja de texto de la opción etiqueta semántica la palabra: Accesorio. En la Figura 3.18 se muestra una imagen del formulario, tras la inserción de los datos:

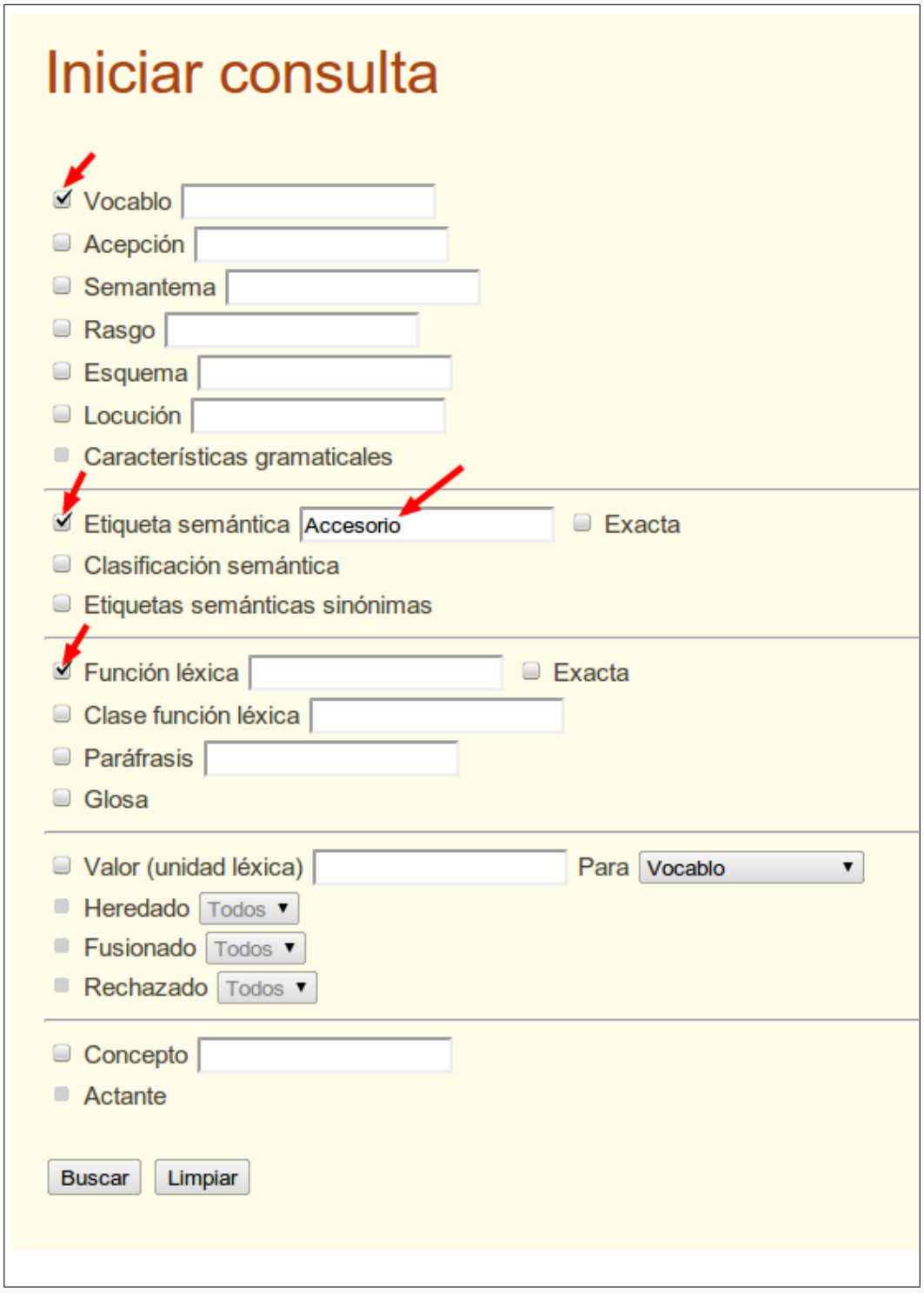


Figura 3.18: *Formulario de consulta*

Una vez seleccionado las opciones convenientes en el formulario, se pulsa el botón «Buscar».

Paso 2

La aplicación al pulsar dicho botón, realiza una consulta a la base de datos de la aplicación, véase Figura 3.19.

```
SELECT DISTINCT fl.expresion as expresion_func,
es.expresion as expresion_semantica,l.vocablo,l.acepcion,
l.categoria_gramatical,l.genero
FROM funcion_lexica fl,etiqueta_semantica es,lema l,
correspondencia_FL_ES_UL fl_es_ul
WHERE es.expresion LIKE "Accesorio%"
AND l.id_ES = es.id_ES
AND fl_es_ul.id_FL = fl.id_FL
AND fl_es_ul.id_ES = es.id_ES
GROUP BY expresion_func, expresion_semantica, l.vocablo
ORDER BY l.vocablo
```

Figura 3.19: *Detalle de la consulta SQL ejecutada para la prueba «Realizar consulta»*

Una vez realizada la consulta, muestra el resultado en una tabla HTML de cuatro columnas: fila, vocablo, etiqueta semántica y función léxica. En la Figura 3.20 se muestra una parte de dicha tabla.

Fila	Vocablo	Etiqueta semántica	Función Léxica
1	abanico	**Accesorio**	FinReal1
2	abanico	**Accesorio**	Degrad
3	abanico	**Accesorio**	Real1
4	abanico	**Accesorio**	IncepReal1
5	abanico	**Accesorio**	AntiFact0
6	abanico	**Accesorio**	PreparFact0
7	agenda	**Accesorio**	Real1
8	agenda	**Accesorio**	IncepReal1
9	agenda	**Accesorio**	AntiFact0
10	agenda	**Accesorio**	PreparFact0
11	agenda	**Accesorio**	FinReal1
12	agenda	**Accesorio**	Degrad

Figura 3.20: *Formulario para iniciar una consulta*

3.6.4. Insertar datos

En la sección de inserción de datos, existe la posibilidad de añadir datos en varias entidades del modelo de la base de datos. Cada una de estas entidades, tiene un formulario de inserción con varias opciones. Por tanto, el número de combinaciones de pruebas es bastante elevado. En la Tabla 3.4 se muestra una aproximación de combinaciones por cada entidad de la base de datos.

Entidad	**Opciones formulario**	**Pruebas generales**	**Pruebas específicas**
Etiqueta semántica	10	9	$2^8 = 128$
Función léxica	19	17	$2^16 = 65536$
Paráfrasis	5	4	$2^3 = 8$
Concepto	8	7	$2^6 = 32$
Unidad léxica	5	4	$2^3 = 8$
Lema	19	19	$2^18 = 262144$
Rasgo	3	3	$2^2 = 4$
Semantema	3	3	$2^2 = 4$
Esquema	3	3	$2^2 = 4$
Locución	3	3	$2^2 = 4$
Clase función léxica	2	1	1
Insertar valores a lema	3	2	2
Insertar valores a etiqueta semántica	3	2	2
Total	86	77	327877

Tabla 3.4: *Combinaciones para formulario de inserción de datos*

La columna «pruebas particulares» de la tabla anterior, calcula las pruebas necesarias si se realizase la combinación de todas las opciones del formulario. Como se puede observar, en algunas entidades es imposible realizar tal cantidad de pruebas, véase entidad lema. Por tanto se ha decidido realizar las pruebas de la columna «pruebas generales». Se cree suficiente dicha batería de pruebas, porque en la programación de la inserción de datos, se ha tratado de forma independiente cada opción de un formulario.

En la Figura 3.21 se muestra el formulario de inserción de la entidad lema, para que se tenga una idea de la amplitud del formulario:

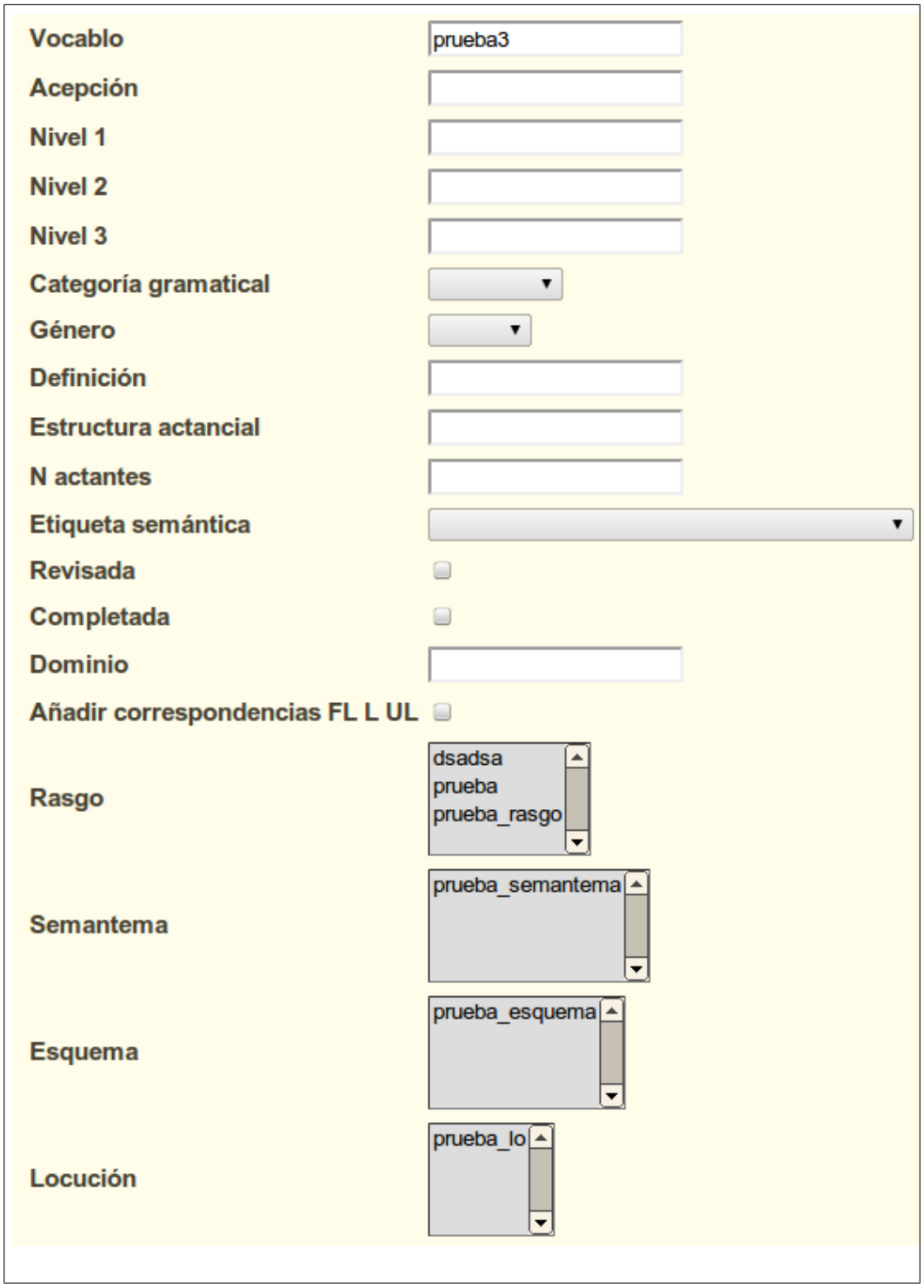


Figura 3.21: *Detalle del formulario de inserción de lema*

3.6.4.1. Prueba insertar datos

El análisis de esta prueba consiste en la inserción de una etiqueta semántica. A continuación, se detallan todos los pasos de la prueba:

Paso 1

Para realizar la inserción de datos, se tiene que estar registrado en la aplicación. Por tanto, en primer lugar se accede a la aplicación, para ello en la parte superior derecha de la aplicación, se introduce el nombre y la contraseña del usuario con el que se quiere entrar.

Paso 2

Una vez realizado el registro, se puede hacer clic en la operación «Insertar datos» del menú. La aplicación muestra un texto de introducción con una lista de entidades en la que se puede insertar datos. Se hace clic en «Etiqueta semántica».

Paso 3

Tras hacer clic, se muestra un formulario con varios campos. En esta prueba, se rellenan algunos de los campos pero en este caso, sólo sería obligatorio rellenar el nombre de la propia etiqueta semántica que se va a insertar. En la Figura 3.22 se muestra el formulario ya relleno:

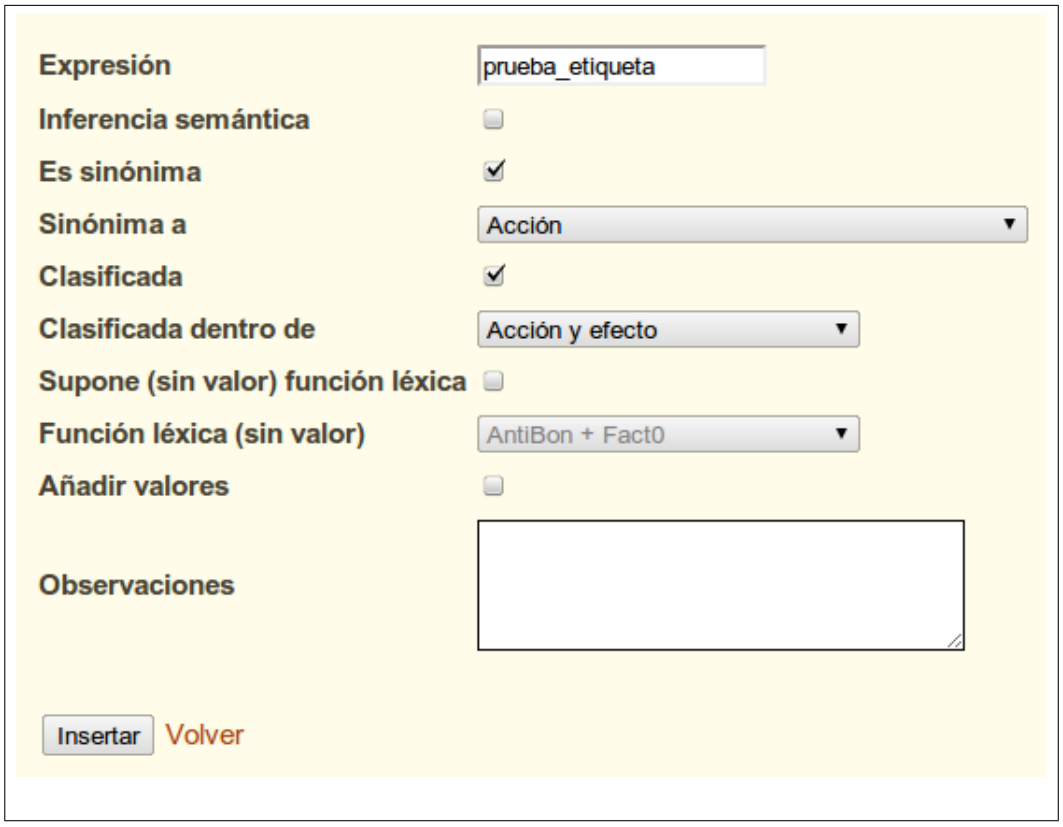


Figura 3.22: *Formulario de inserción de etiqueta semántica*

Una vez relleno el formulario, se hace clic en el botón «Insertar»

Paso 4

Para realizar la inserción de los datos, la aplicación ejecuta varias sentencias SQL. En concreto, las siguientes:

```
REPLACE INTO
etiqueta_semantica (idioma,id_es,expresion,
inferencia_semantica,autor)
VALUES ('1','646','prueba_etiqueta','0','admin')
```

Después inserta la etiqueta semántica recién creada en la tabla de etiquetas sinónimas:

```
REPLACE INTO ES_sinonima (id_es1,es_sinonima,autor)
VALUES ('646','Acción','admin')
```

Y por último, inserta la etiqueta en la tabla de clasificación de etiquetas:

```
REPLACE INTO clasificacion_de_ES (id_es1,id_es2,autor)
VALUES ('503','646','admin')
```

La aplicación muestra un mensaje informando de que la operación se ha realizado con éxito.

3.6.5. Modificar datos

En la sección de modificar datos, existe la posibilidad de modificar datos sobre varias entidades del modelo de la base de datos. Cada una de estas entidades, tienen un formulario de modificación con varias opciones. La batería de pruebas de esta sección, es más complicada porque el número de opciones del formulario de modificación, depende en gran medida de si el elemento elegido tiene dependencias. Por ejemplo, si se elige el elemento «Abdomen», se aprecia que tiene dependencia con la entidad «Clasificación de etiqueta semántica» del modelo. En la Figura 3.23 se muestra este caso:

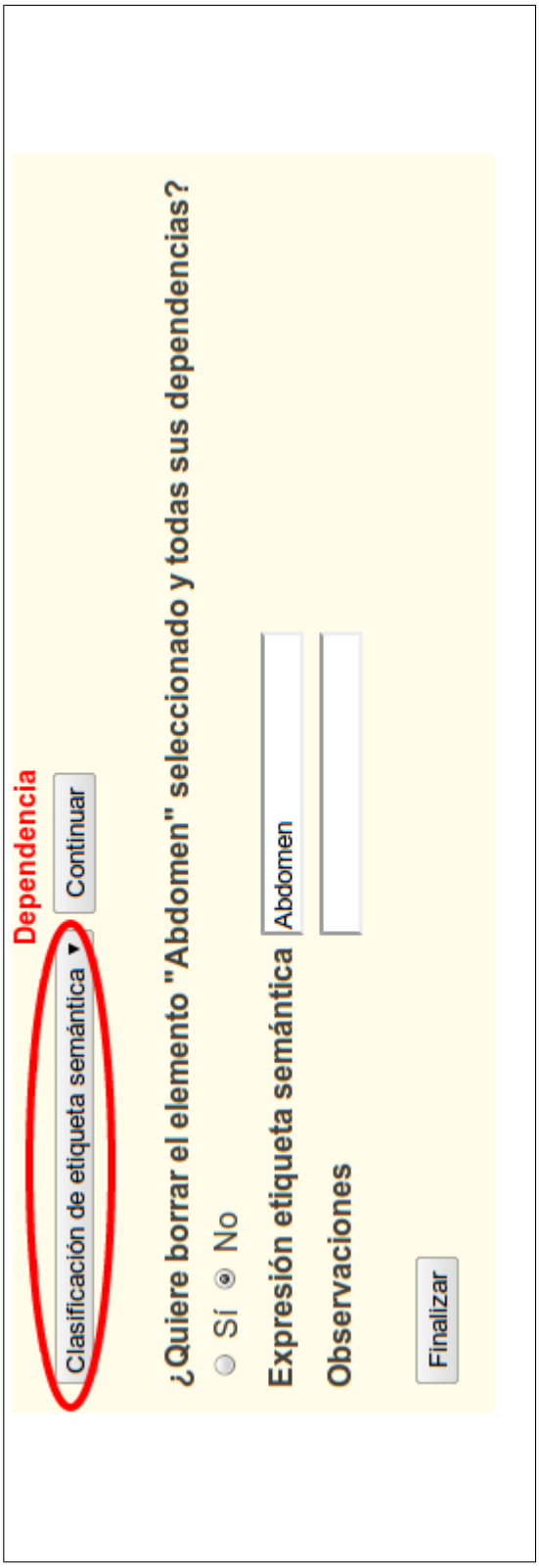


Figura 3.23: *Detalle de dependencia de la etiqueta «Abdomen»*

Por tanto, según las dependencias de un elemento las opciones del formulario cambian.

La solución que se ha llevado a cabo es crear/encontrar un elemento por cada formulario de modificación, que tenga todas las dependencias posibles y realizar las pruebas con dicho elemento. De esta manera, la batería de pruebas queda de la siguiente manera:

Entidad	**Opciones**	**Dependencias**	**Suma**
Etiqueta semántica	1	6	7
Función léxica	3	4	7
Paráfrasis	2	3	5
Concepto	3	1	4
Unidad léxica	5	1	6
Lema	6	3	9
Rasgo	1	1	2
Semantema	1	1	2
Esquema	1	1	2
Locución	1	1	2
Clase función léxica	1	1	2
Modificar valores a lema	1	0	1
Modificar valores a etiqueta semántica	1	0	1
Total de pruebas	27	23	50

Tabla 3.5: *Combinaciones para el formulario de modificación de datos*

Como se especifica en la tabla anterior, la batería de pruebas de esta sección se compone de 50 pruebas en total.

3.6.5.1. Prueba modificar datos

Esta prueba consiste en la modificación de una etiqueta semántica que tiene dependencias. A continuación, se detallan todos los pasos de la prueba:

Paso 1

Para realizar la modificación de datos, se tiene que estar registrado en la aplicación. Por tanto, en primer lugar se accede a la aplicación, para ello en la parte superior derecha de la aplicación, se introduce el nombre y la contraseña del usuario con el que se quiere entrar.

Paso 2

Una vez realizado el registro, se puede hacer clic en la operación «Modificar datos» del menú. La aplicación muestra un texto de introducción con una lista de entidades en la que se puede modificar datos. Se hace clic en «Etiqueta semántica».

Paso 3

Tras hacer clic, se muestra un formulario donde se elige aquella etiqueta que se quiere modificar, en este ejemplo se elige «Abdomen». Después se pulsa el botón «Siguiente».

Paso 4

Una vez elegida dicha etiqueta semántica, se muestra dos formularios. El

primero de ellos, relacionado con la dependencia de la etiqueta «Abdomen» con la clasificación de etiquetas semánticas. El segundo de los formularios, permite modificar el nombre de la propia etiqueta semántica.

Paso 5

En primer lugar se modifica la relación de la etiqueta con la clase de etiquetas semánticas. Para ello, se pulsa el botón «Continuar» y automáticamente aparecerá un nuevo formulario donde la aplicación ofrece cambiar la clase de etiqueta semántica de la etiqueta elegida. En este ejemplo, se modifica la clase «Parte del abdomen» por «Parte de un animal». Para que la modificación, sea efectiva se pulsa en el botón «Validar». En la Figura 3.24 se muestra esta situación:

Figura 3.24: *Detalle modificación de clasificación de etiqueta semántica*

Paso 6

Tras modificar la clase de la etiqueta semántica, se modifica el nombre de la misma. Por lo tanto, en la caja de texto de «Expresión de la etiqueta semántica» se escribe el nuevo nombre de la etiqueta. Una vez escrito, se pulsa el botón «Finalizar». Para realizar la inserción de los datos, la aplicación ejecuta una sentencia SQL. En concreto, la siguiente:

```
UPDATE etiqueta_semantica
SET expresion = 'Nuevo_Abdomen',
autor = 'admin',
observaciones = '',
fecha = '2011-08-29 00:28:02'
WHERE id_es = '517'
```

La aplicación muestra un mensaje informando de que la operación se ha realizado con éxito.

3.6.6. Insertar usuario

La inserción de un usuario es una operación sencilla. Su funcionamiento se verifica con una única prueba.

3.6.6.1. Prueba insertar usuario

Esta prueba consiste en la inserción de un usuario de prueba. A continuación, se detallan todos los pasos de la prueba:

Paso 1

Para realizar la inserción de un usuario, se debe estar registrado en la aplicación. Por tanto, en primer lugar se accede a la aplicación, para ello en la parte superior derecha de la aplicación, se introduce el nombre y la contraseña del usuario con el que se quiere entrar.

Paso 2

Una vez realizado el registro, se puede hacer clic en la operación «Administrar usuarios» del menú. La aplicación muestra un texto de introducción con una lista de operaciones, relacionadas con la administración de usuarios. Se hace clic en «Insertar usuario».

Paso 3

Tras hacer clic, se muestra un formulario donde se debe introducir nombre, contraseña y correo electrónico del usuario. Si el usuario que se quiere insertar, tiene que tener permisos de administrador, se debe hacer clic en la casilla «Administrador». Una vez introducido los datos, se pulsa el botón «Insertar». En la Figura 3.25 se detalla esta situación:

Figura 3.25: *Formulario inserción de usuario*

Para realizar la inserción del nuevo usuario, la aplicación ejecuta la siguiente sentencia SQL:

```
INSERT INTO sf_guard_user
(algorithm, is_active, is_super_admin, username,email_address,
salt, password, created_at, updated_at)
VALUES ('sha1', '1', '0', 'prueba_usuario','prueba@gmail.com',
'7e7bebc6fe53d5bf1c9f48206a5ba3ed',
'1c8a421beb002b127acce97c754d1838563c5938',
'2011-08-29 00:37:25', '2011-08-29 00:37:25')
```

3.6.7. Eliminar usuario

La eliminación de un usuario es una operación sencilla. Su funcionamiento se verifica con una única prueba.

3.6.7.1. Prueba eliminar usuario

Esta prueba consiste en la eliminación de un usuario de prueba. A continuación, se detallan todos los pasos de la prueba:

Paso 1

Para eliminar un usuario, se debe estar registrado en la aplicación. Por tanto, en primer lugar se accede a la aplicación, para ello en la parte superior derecha de la aplicación, se introduce el nombre y la contraseña del usuario con el que se quiere entrar.

Paso 2

Una vez realizado el registro, se puede hacer clic en la operación «Administrar usuarios» del menú. La aplicación muestra un texto de introducción con una lista de operaciones, relacionadas con la administración de usuarios. Se hace clic en «Eliminar usuario».

Paso 3

Tras hacer clic, se muestra un desplegable donde se muestran todos los usuarios registrados en la aplicación. Se elige el usuario que se quiere eliminar. Una vez elegido el usuario, se pulsa el botón «Eliminar». En la Figura 3.26 se incorpora una imagen de esta situación:

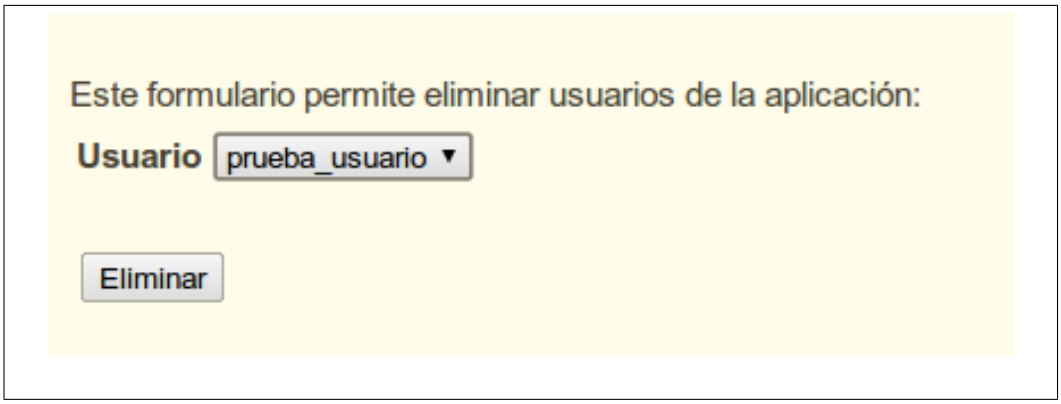


Figura 3.26: *Formulario eliminación de usuario*

Para realizar la eliminación del usuario, la aplicación ejecuta una sentencia SQL. En concreto, la siguiente:

```
DELETE
FROM sf_guard_user
WHERE id = '3'
```

La aplicación muestra un mensaje informando de que la operación se ha realizado con éxito.

3.6.8. Mostrar usuarios

Mostrar todos los usuarios de la aplicación es una operación sencilla. Su funcionamiento se verifica con una única prueba.

3.6.8.1. Prueba mostrar usuarios

Esta prueba consiste en mostrar todos los usuarios de la aplicación. A continuación, se detallan todos los pasos de la prueba:

Paso 1

Para mostrar los usuarios de la aplicación, se tiene que estar registrado en la aplicación. Por tanto, en primer lugar se accede a la aplicación, para ello en la parte superior derecha de la aplicación, se introduce el nombre y la contraseña del usuario con el que se quiere entrar.

Paso 2

Una vez realizado el registro, se puede hacer clic en la operación «Administrar usuarios» del menú. La aplicación muestra un texto de introducción con una lista de operaciones, relacionadas con la administración de usuarios. Se hace clic en «Mostrar usuarios».

Paso 3

Tras hacer clic, se muestra una tabla HTML con tres columnas: id del usuario en el sistema, nombre y correo electrónico del usuario. En la Figura 3.27 se muestra esta situación:

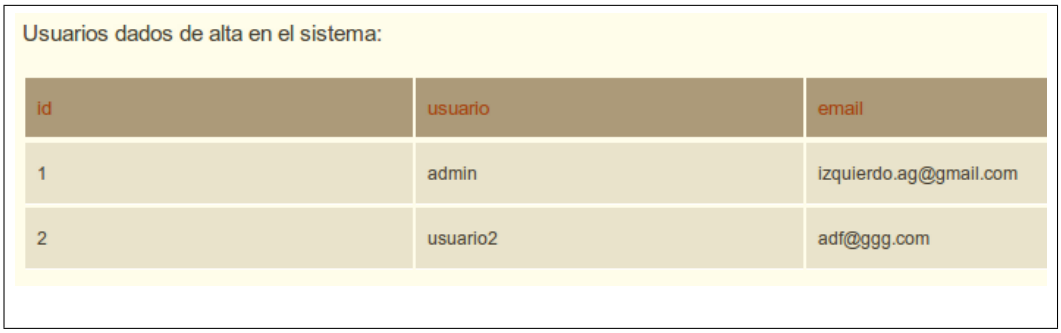


Figura 3.27: *Tabla de usuarios registrados en el sistema*

Para mostrar todos los usuarios del sistema, la aplicación ejecuta una sentencia SQL. En concreto, la siguiente:

```
SELECT username,email_address,id
FROM sf_guard_user
```

3.6.9. Realizar copia de seguridad

Existen dos posibilidades, que se quiera guardar la copia de seguridad en el servidor de la aplicación o que se quiera descargar y guardar localmente. Basta con realizar dos pruebas para verificar que la operación funciona correctamente.

3.6.9.1. Prueba realizar copia de seguridad

Esta prueba consiste en realizar una copia de seguridad y guardar el fichero en una carpeta del servidor. A continuación, se detallan todos los pasos de la prueba:

Paso 1

Para realizar la copia de seguridad, se tiene que estar registrado en la aplicación. Por tanto, en primer lugar se accede a la aplicación, para ello en la parte superior derecha de la aplicación, se introduce el nombre y la contraseña del usuario con el que se quiere entrar.

Paso 2

Una vez realizado el registro, se puede hacer clic en la operación «Administrar BD» del menú. La aplicación muestra un texto de introducción con una lista de operaciones, relacionadas con la administración de la base de datos. Se hace clic en «Realizar copia de seguridad».

Paso 3

Tras hacer clic, se muestra un formulario. En primer lugar, en el desplegable se selecciona la opción «servidor». Después se introduce el nombre del archivo, y la ubicación donde la aplicación guardará el fichero generado. La ubicación debe ser completa y con permisos de escritura. Una vez relleno el formulario, se hace pulsa el botón «Realizar backup». En la Figura 3.28 se muestra esta situación:

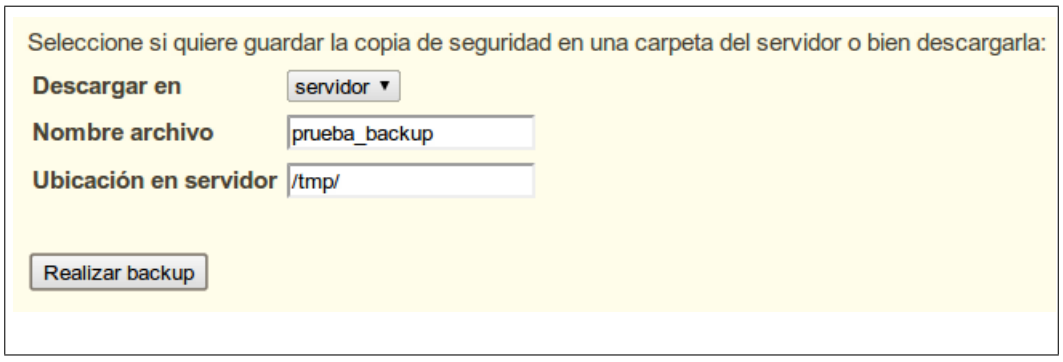


Figura 3.28: *Formulario para realizar una copia de seguridad*

La aplicación muestra un mensaje informando de que la operación se ha realizado con éxito.

3.6.10. Cargar copia de seguridad

Existen dos posibilidades, que se quiera cargar la copia de seguridad desde un fichero alojado en el servidor de la aplicación o que se quiera cargar desde un fichero local. Basta con realizar dos pruebas para verificar que la operación funciona correctamente.

3.6.10.1. Prueba realizar carga de una copia de seguridad

Esta prueba consiste en cargar una copia de seguridad alojada en el servidor de la aplicación. A continuación, se detallan todos los pasos de la prueba:

Paso 1

Para cargar la copia de seguridad, se tiene que estar registrado en la aplicación. Por tanto, en primer lugar se accede a la aplicación, para ello en la parte superior derecha de la aplicación, se introduce el nombre y la contraseña del usuario con el que se quiere entrar.

Paso 2

Una vez realizado el registro, se puede hacer clic en la operación «Administrar BD» del menú. La aplicación muestra un texto de introducción con una lista de operaciones, relacionadas con la administración de la base de datos. Se hace clic en «Cargar copia de seguridad».

Paso 3

Tras hacer clic, se muestra un formulario. En primer lugar, en el desplegable se selecciona la opción «servidor». Después se introduce la ubicación completa donde está el fichero de la copia de seguridad. Una vez introducida la ubicación, se pulsa el botón «Cargar backup». En la Figura 3.29 se muestra esta situación:

Figura 3.29: *Formulario para cargar una copia de seguridad*

Prueba	Entrada	Salida esperada	Salida obtenida
Realizar consulta	Vocablos y funciones léxicas de la etiqueta semántica <<Accesorio>>	Tabla de cuatro columnas con los vocablos y funciones léxicas de la etiqueta semántica <<Accesorio>>	Salida obtenida igual a la esperada
Insertar datos	Etiqueta semántica sinónima a <<Acción>> clasificada dentro de <<Acción y efecto>>	Mensaje de que la inserción se ha realizado correctamente	Salida obtenida igual a la esperada
Modificar datos	Nueva clase y nombre de la etiqueta semántica <<Abdomen>>	No obtener ningún mensaje de error	Salida obtenida igual a la esperada
Insertar usuario	Nombre de usuario, contraseña y correo electrónico del usuario a insertar	Mensaje de que la inserción se ha realizado correctamente	Salida obtenida igual a la esperada
Eliminar usuario	Nombre del usuario que se dará de baja	Mensaje de que la operación se ha realizado con éxito	Salida obtenida igual a la esperada
Mostrar usuarios	Sin entrada	Tabla de los usuarios dados de alta en el sistema	Salida obtenida igual a la esperada
Realizar copia de seguridad	Nombre del archivo, descargar en el servidor, ubicación dentro del servidor	Mensaje de que la operación se ha realizado con éxito	Salida obtenida igual a la esperada
Cargar copia de seguridad	Cargar copia desde el servidor, ubicación del archivo dentro del servidor	Mensaje información de que la copia de seguridad se ha cargado con éxito	Salida obtenida igual a la esperada

Figura 3.30: *Detalle de la entrada, salida esperada y obtendida de las pruebas realizadas*

3.6.11. Pruebas de validación

El objetivo de este tipo de pruebas no es encontrar fallos del producto, se diseñan para verificar que se han cumplido los requisitos de análisis[50].

Estas pruebas de validación, se realizaron cada vez que una sección de la aplicación quedaba terminada. Además, una vez terminada todas las secciones del producto, se realizaron una serie de pruebas de validación que engloban a toda la aplicación.

Muchas de las pruebas de validación han sido realizadas por el propio usuario, gracias a que fue posible una retroalimentación fluida entre el cliente y el desarrollador de la aplicación. Las pruebas se realizaron en ciclos cortos minizando los cambios que no cumplen con los requisitos[15][51].

Para aclarar qué módulo o módulos cubre cierto requisito, se ha realizado una tabla en donde se relacionan todos los módulos del sistema y los requisitos funcionales definidos en la sección 3.3, véase la Figura 3.31.

[50]Manuel Collado. http://lml.ls.fi.upm.es/ep/0809/pruebas.pps
[51]Beck, Kent. 1999. Extreme Programming Explained: Embrace Change

	Index	Consultas	Insertar	Modificar	Usuarios	Dump	Export	Download
Req01		X						
Req02		X						
Req03		X						
Req04		X						
Req05		X						
Req06		X						
Req07		X					X	
Req08			X					
Req09			X					
Req10			X					
Req11				X				
Req12				X				
Req13				X				
Req14				X				
Req15	X							
Req16	X							
Req17	X							
Req18					X			
Req19					X			
Req20					X			
Req21						X		
Req22						X		X
Req23						X		
Req24						X		
Req25						X		
Req26	X							
Req27	X							

Figura 3.31: *Matriz de trazabilidad de requisitos y módulos*

3.6.12. Pruebas de regresión

Se denominan pruebas de regresión a aquellas pruebas que intentan descubrir errores causados a cambios en la aplicación, como por ejemplo arreglar otro tipo de error. Esto implica que el error aparece de forma inesperada debido a un cambio de la aplicación.

Para evitar que este tipo de error ocurriera, cada vez que se terminaba de arreglar un error, se lanzaban la batería de pruebas unitarias mediante el *framework* Symfony.

3.7. Ejemplo de desarrollo completo de un módulo del sistema

Esta sección describe cuál fue el proceso completo de desarrollo y toma de decisiones de un módulo del sistema. De esta manera, se podrá observar que a partir de unas funcionalidades dadas por el cliente, se llega a un módulo final, integrado en el sistema.

3.7.1. Diseño

Mediante el documento de Especificación de Requisitos Software, en adelante ERS, se analizaron aquellos requisitos funcionales que tenían características en común y que no se habían implementado. Se observó que los requisitos: Req(16), Req(17), Req(18), Req(19), Req(20), Req(21) se encargaban de toda la administración de los usuarios de la aplicación. Según el documento ERS, todos estos requisitos tienen prioridad alta, por lo tanto, tienen que cumplirse sin discusión alguna.

Por lo tanto, se pensó que lo más conveniente sería realizar un módulo que agrupase toda la funcionalidad relacionada con la administración de usuarios. A este módulo se le llamó «usuarios». Todos los módulos del sistema, están formados por tres carpetas: *actions*, *templates* y *lib* correspondientes al controlador, vista y funciones auxiliares que puede utilizar el módulo.

A partir de los requisitos comentados anteriormente, se analizó cómo debían ser las vistas del módulo, es decir, las diferentes páginas web que tendría la sección de administración. Después del análisis previo, se desarrollaron las

vistas en archivos de tipo PHP, donde está encapsulado código HTML.

- indexSuccess.php. En esta vista se muestran las operaciones que un usuario autorizado puede realizar dentro de la sección de administración de usuarios. No recibe ningún dato por parte del controlador.
- addUserSuccess.php. Si el usuario pulsa en la operación de añadir usuario, se mostrará esta vista, Figura 3.32. En este caso, la vista recibe una variable de tipo formulario del controlador, para que pueda mostrar el formulario de inserción de usuario.
- newUserSuccess.php. Se informa si la operación de inserción de usuario se ha realizado correctamente o no, a partir de una variable que el controlador le ha mandado previamente, Figura 3.33.
- listUserSuccess.php. Para seleccionar el usuario que se quiere eliminar, se muestra un formulario con un desplegable donde aparecen todos los usuarios del sistema. Esta lista de usuarios, la manda el controlador.
- deleteUserSuccess.php. Se informa si la eliminación del usuario se ha realizado con éxito o no, a partir de la información que el controlador le ha mandado previamente.
- usersSuccess.php. Esta vista, se encarga de mostrar la lista de usuarios dados de alta en el sistema, que el controlador le ha mandado previamente.

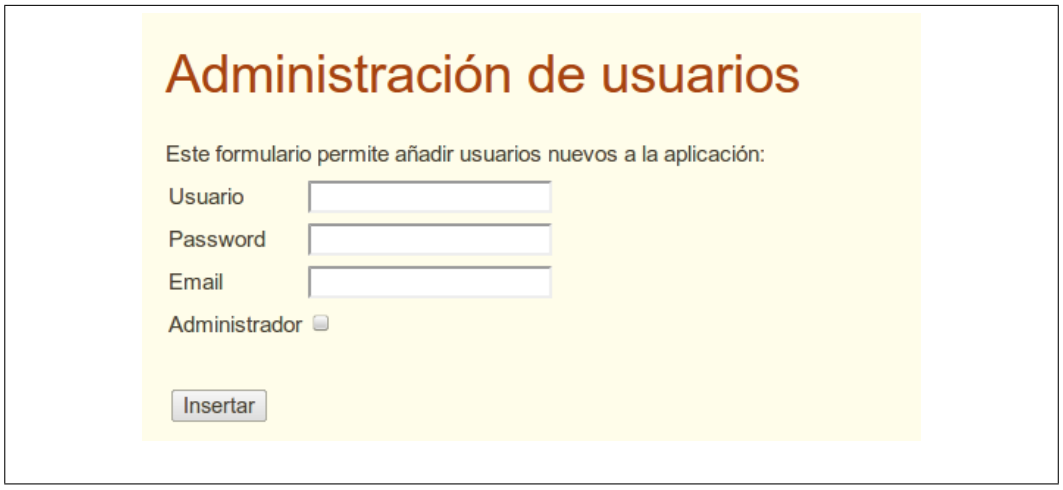


Figura 3.32: *Formulario para añadir un nuevo usuario en la aplicación*

Figura 3.33: *Formulario para eliminar un usuario de la aplicación*

Todos estos archivos PHP son las vistas del módulo, por lo tanto, se guardan en la carpeta *templates*.

Una vez que se sabe cuál es la interfaz de la sección de administración de usuarios, se analiza qué funciones o procedimientos hay que desarrollar en el controlador del módulo para que las vistas muestren los datos correctamente. Es importante recalcar, que en este caso, es necesario implementar ciertas funciones en el controlador del módulo porque las vistas descritas

anteriormente modifican y obtiene datos del modelo, es decir, de la base de datos.

Funciones implementadas en el controlador del módulo:

- *executeAddUser*. Esta función crea el formulario de inserción de usuario que más tarde utilizará la vista para presentarlo en formato HTML.
- *executeNewUser*. La función recupera los campos del formulario de inserción de usuario: nombre de usuario, contraseña, email, administrador (si/no). Se inserta en la base de datos de la aplicación y se informa a la vista si todo se ha realizado correctamente o no.
- *executeListUser*. Para la eliminación de un usuario, es necesario listar todos los usuarios dados de alta en la aplicación. Esta función llama a una función auxiliar *getUsuarios* que devuelve un *array* multidimensional con el id, nombre y correo electrónico de cada usuario. Teniendo la información de todos los usuarios, la función *executeListUser* podrá generar el formulario de eliminación de usuario.
- *executeDeleteUser*. Recupera los campos del formulario de eliminación de usuario. Se elimina el usuario de la base de datos de la aplicación y se informa a la vista si la operación se ha realizado con éxito o no.
- *executeUsers*. Se encarga de mostrar todos los usuarios del sistema en una tabla HTML. Para ello, se ayuda de la función auxiliar *getUsuarios*, mencionada anteriormente, que le proporciona los usuarios dados de alta en la aplicación. Esta información se manda a la vista para que muestre correctamente la tabla de usuarios.

Ya que el controlador utiliza en dos ocasiones una consulta para obtener la lista de usuarios del sistema, se implementó una función auxiliar *getUsuarios* dentro de la carpeta *lib/* en un archivo llamado *myUsuarios.class.php* que se corresponde con una clase PHP. Es obligatorio que si el fichero se llama *myUsuarios.class.php* la clase PHP se llame *myUsuarios*.

3.7.2. Implementación

En esta fase se implementa el módulo «usuarios». Todos los módulos del Sistema, están formados por tres carpetas: actions, templates y lib correspondientes al controlador, vista y funciones auxiliares que puede utilizar el módulo. Las vistas son archivos de tipo PHP, donde está encapsulado código HTML.

3.7.3. Pruebas

En esta sección, se va a explicar el proceso de pruebas que se realizó para intentar encontrar errores en el desarrollo del módulo «usuarios».

En primer lugar, se pensó que se debía de realizar una serie de pruebas de caja negra sobre el módulo. Por lo tanto, se diseñaron una batería de entradas, en las que se intentó que fuesen poco usuales, con el objetivo de encontrar la máxima cantidad de errores posibles. Cuando el diseño de estas entradas terminó, se inició la ejecución de las pruebas de caja negra.

Cada vez que una prueba de caja negra no ofrecía los resultados esperados, se paraba la ejecución de las pruebas, se buscaba y se corregía el error. Si uno de estos errores, se había producido en la función auxiliar *getUsuarios*,

se diseñaba una prueba unitaria mediante la forma que aconseja Symfony, véase capítulo 3.6.1.

Una vez finalizadas las pruebas de caja negra, comenzaron las pruebas de validación. Al realizar estas pruebas con los usuarios de la aplicación, a parte de verificar que se cumplen los requisitos de análisis especificados en el documento ERS, también se analizó que la interfaz de la sección sea del gusto de los usuarios. El único cambio que se introdujo, fue que en la página que muestra todos los usuarios dados de alta, no se mostraba el email de cada uno de ellos. Esto fue un error en fase de implentanción al crear la vista de la página, ya que en el controlador se había tenido en cuenta. Por lo tanto, tan sólo se tuvo que modificar la vista del módulo que afectaba a esa parte (*usersSuccess.php*). En este tipo de errores, es donde uno se da cuenta que tener separada la vista del controlador ayuda muchísimo en la depuración de código. Todo gracias al uso del patrón modelo-vista-controlador.

Terminadas todas las pruebas previstas, se dio por valido el módulo «usuarios».

Capítulo 4

Conclusiones y trabajo futuro

Para finalizar esta memoria, se van a plasmar unas conclusiones sobre el proyecto realizado.

El Trabajo Fin de Carrera que aquí se ha presentado ha consistido en el estudio de un modelo lingüístico y el desarrollo de una aplicación web para consultar e insertar datos sobre el mismo. Un análisis simplista podría llevar a pensar que se trata de una tarea sencilla pues, al fin y al cabo, en términos de producto lo que se obtiene es una aplicación web que realiza consultas que extraen e insertan datos. Sin embargo, ha supuesto un buen reto, ya que se ha tenido que desentrañar las múltiples consultas que el modelo ofrecía. También se invertió mucho tiempo en buscar la mejor opción en el ámbito de la interfaz web, ya que uno de los requisitos era desarrollar aplicación sencilla, usable y dirigida para todo tipo de usuarios, ya sean curiosos o lingüistas.

Se ha intentado en todo momento que la codificación en los lenguajes HTML y CSS, fuesen lo más sencillas y correctas posibles, para que las diferentes interpretaciones de los navegadores web, no afectase la interfaz del producto. El nexo de unión del modelo de datos con la vista, ha sido el

lenguaje PHP. Además, se ha usado JavaScript para dar cierto dinamismo a la aplicación. El resultado ha sido una aplicación sin excesos decorativos, pero suficientemente clara para que la navegación por la misma sea amigable.

Los tiempos de respuesta del producto son, en términos generales, correctos. Aunque el usuario puede realizar consultas que presentan una fuerte carga computacional, los tiempos no son excesivos. Por ejemplo, una de las consultas más complicadas es: consultar la clasificación de las etiquetas semánticas y vocablos del modelo, el tiempo que invierte la aplicación en presentar la información pedida, es aproximadamente de cuatro segundos.

En el estudio previo del proyecto, se presumía que la parte más complicada sería la modificación e inserción de datos. Pero con el proyecto ya finalizado, se ha comprobado que la dificultad principal ha sido codificar todas las posibles consultas que el usuario puede realizar. Gran parte de esta estimación errónea, se debe a la decisión de utilizar jQuery, una biblioteca de JavaScript. Esta decisión ayudó a reducir el número de páginas de la aplicación, haciendo que los formularios de inserción y modificación de datos fuesen más dinámicos. Otra de las decisiones que ocasionaron que la estimación del tiempo empleado en la sección de consultas, fuese errónea, se debió al hecho de tener que rehacer prácticamente el código de dicha sección. En un principio, se pensó en ir concatenando las opciones que el usuario elegía en una única consulta, pero su depuración era muy complicada. Por lo tanto, se decidió tratar cada consulta de manera independiente.

Aunque el producto cumple todos los requisitos establecidos, eso no significa que la aplicación no sea susceptible de mejoras que permitan optimizar la aplicación y subsanen posibles errores descubiertos en el futuro, y de nuevas funcionalidades que den respuesta a las diferentes necesidades que han

surgido o vayan surgiendo en el futuro. Así, las líneas futuras del desarrollo de la aplicación podrían encaminarse hacia distintos aspectos, entre los que se pueden señalar los siguientes:

- Aumentar el número de consultas que el usuario puede realizar. Por ejemplo: mostrar el concepto de cierta función léxica.
- Contemplar varias lenguas. El modelo conceptual de la base de datos es fácilmente adaptable a cualquier idioma, lo que permitiría tener bases de datos léxicas correspondientes a distintos idiomas. La posibilidad de que la herramienta permita trabajar con estas bases de datos simultáneamente sería muy útil para hacer estudios comparativos y, especialmente, para traducción.
- Actualmente la aplicación muestra los datos en formato tabla, pero quizás para ciertas consultas sea conveniente una representación más gráfica. Por ejemplo, si se quiere mostrar la clasificación de etiquetas semánticas, es posible que una representación en forma de árbol sea más adecuada y didáctica. También sería muy útil ver de manera gráfica una relación entre unidades léxicas mediante funciones léxicas.
- Emplear un modelo de usuario que permita adaptar la herramienta a las características de quién la emplee. Por ejemplo, para que cada usuario pueda guardar una serie limitada de consultas «favoritas».
- Para controlar la inserción y modificación de los datos, se podría implementar una funcionalidad para los usuarios administradores que permita conocer fácilmente quién ha realizado los últimos cambios en la BD y en qué momento.

Por último, me gustaría señalar que el trabajo en esta memoria también ha sido una labor muy importante y educativa, pues ha ayudado a analizar y estructurar el conocimiento adquirido durante este trabajo, y a apreciar la importancia de una redacción cuidada.

Bibliografía y referencias

[1] Barrios, María A., Bernardos, M. Socorro. *BaDELE3000. An implementation of the lexical inheritance principle*, Wiener Slawistischer Almanach, Sonderband 69. 2007.

[2] Barrios, María A. *El dominio de la funciones léxicas en el marco de la Teoría Sentido-Texto*, Tesis doctoral. Estudios de Lingüística del español (ELiEs), vol 30. 2010.

[3] Mel'cuk, Igor A. *Meaning-Text Models: A recent trend in Soviet linguistics*, Annual Review of Anthropology 10: 2762. 1981

[4] Goossens, Michel. *Diccionario de Organización del Conocimiento*, `http://www.eubca.edu.uy/diccionario/`. Consultado en septiembre de 2011.

[5] Mercedes. *Teoría de la valencia y concepto de actante*, `http://espanolenamerica.wordpress.com/`. 2010.

[6] Observatoire de linguistique Sens-Texte (OLST). *DiCoDOC. Le DiCo et sa version DiCouèbe*, `http://http://olst.ling.umontreal.ca/dicouebe/`. 2005.

[7] Barry W. Boehm. *A Spiral Model of Software Development and Enhancement*, ACM SIGSOFT Software Engineering Notes, Volume 21 Issue 5. 1986.

[8] Cood, E.F. *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, Volume 13 Issue 6. 1970.

[9] Chen, Peter. *The Entity Relationship Model - Toward A Unified View of Data*, ACM Transactions on Database Systems, Special Issue, Volume 1 Issue 1. 1975

[10] jQuery . *Documentación biblioteca jQuery*, `http.//docs.jquery.com/Main_Page`. Consultado en 2011.

[11] Potencier, Fabien . *Practical Symfony*, 1.4 Version. 2009

[12] Symfony. *Blog oficial de Symfony*, `http://symfony.com/blog/`. Consultado en 2011

[13] DiCouèbe . *Diccionario en línea de combinatoria de lengua francesa*, `http://olst.ling.umontreal.ca/dicouebe/main.php`. Consultado en 2011.

[14] DiCoDOC. *Le DiCo et sa version DiCouèbe*, `http://olst.ling.umontreal.ca/dicouebe/DiCoDOC.pdf`. 2005.

[15] Beck, Kent. *Extreme Programming Explained: Embrace Change*. Addison-Wesley. 1999

[16] Real Academia Española. *Diccionario de la Real Academia Española*, `http.//www.rae.es/rae.html`. Consultado en 2011.

[17] The PHP Group. *Historia de PHP*, `http://es2.php.net/manual/es/history.php`. Consultado en 2011.

[18] Wikipedia. *Ciclo de vida en cascada*, `http://en.wikipedia.org/wiki/Waterfall_model`. Consultado en 2011.

[19] Wikipedia. *Ciclo de vida en espiral*, `http://en.wikipedia.org/wiki/Spiral_model`. Consultado en 2011.

[20] Tavorelli, Dario. *The Beauty of LaTeX*, `http://nitens.org/taraborelli/latex`. 2011.

Apéndice A

Manual de instalación

A.1. Introducción

En esta sección se presenta el Manual de Instalación del sistema. En él se detalla todo el proceso de instalación. Como se podrá observar más adelante, este manual está dirigido para sistemas de tipo Unix. Los usuarios que quieran implantar el sistema en un entorno Microsoft Windows, deberán realizar las mismas operaciones pero con la sintaxis propia del sistema.

En primer lugar se especifican los requisitos mínimos necesarios para que el sistema funcione correctamente. A continuación, se describen las operaciones necesarias para implantar el sistema.

A.2. Requisitos del sistema

Los requisitos del sistema son los mismos que necesita el propio *framework* Symfony y que se especifican en su página web.

En primer lugar, es necesario un servidor web. El sistema se desarrolló con el servidor web Apache, que dispone de licencia GPL *GNU General Public License*.

En segundo lugar, es necesario un sistema gestor de bases de datos. Como se ha comentado anteriormente, Symfony 1.4[52] incorpora una capa de abstracción de la base de datos llamado Doctrine. Por tanto, el tipo del sistema gestor de la base de datos también es un requisito que se deja a elección del usuario, aunque se recomienda el uso del gestor MySQL, ya que es el gestor que se utilizó para el desarrollo del sistema.

El siguiente de los requisitos es tener instalado una versión del lenguaje de programación PHP, en concreto, 5.2.4 o superior. En sistemas Unix se puede obtener la versión PHP mediante el comando:

```
$ php -v
\end{figure}
\end{itemize}
Ejemplo de salida del comando anterior:
\begin{figure}[H]
\begin{Verbatim}
$ php -v
PHP 5.3.2-1ubuntu4.9 with Suhosin-Patch (cli)
(built: May  3 2011 00:43:34)
Copyright (c) 1997-2009 The PHP Group
```

[52]http://www.symfony-project.org/installation/1_4

A.3. Instalación

Después de haber reunido los requisitos previos, se puede realizar una instalación correcta del sistema. Para comenzar la instalación se debe descomprimir el código fuente de la aplicación en una carpeta donde el servidor web pueda acceder. Para ello utilizamos el comando:

```
$ tar -xvz dblingua-1.0.0.tgz
```

Es importante recordar que una vez realizada la instalación, se debe configurar el servidor Apache de una manera correcta para que se acceda correctamente a la ubicación donde se acaba de descomprimir el fichero (Véase apartado Configuración del servidor Apache). Después de la extracción se habrá creado un árbol de directorios. A continuación se explica el significado de todas las carpetas del producto:

- apps. En esta carpeta se encuentran todas las aplicaciones del producto.
- cache. Se encuentran los archivos de cache de la aplicación.
- config. Archivos de configuración del proyecto.
- data. Se almacenan los archivos relacionados con los datos. Por ejemplo, el esquema de una base de datos, el archivo que contiene las instrucciones SQL para crear la estructura de tablas, etc.
- doc. En esta carpeta se guarda la presente documentación, el diagrama entidad-relación de la base de datos, scripts de creación y carga del modelo de datos.

- lib. Dentro de esta carpeta se encuentra el propio *framework* Symfony 1.4, más concretamente en *lib/vendor/symfony*. Además se encuentran las bibliotecas y clases del propio proyecto.
- log. En esta carpeta se encuentran los archivos de registro.
- plugins. En esta carpeta se encuentran los plugins que se hayan instalado en el sistema:
  - sfDoctrineGuardPlugin. Es un plugin que ayuda en la autentificación de usuarios para asegurar diferentes partes de la aplicación. |item sfFormExtraPlugin. Este plugin se mantiene por los desarrolladores de Symfony y permite la creación de formularios más completos y seguros.
  - sfPhpExcelPlugin. Mediante este plugin se puede realizar exportaciones de datos a ficheros de tipo Microsoft Office Excel o PDF.
- test. Los archivos de pruebas unitarias y funcionales.
- web. El directorio raíz web

Como se ha explicado anteriormente, en la carpeta *lib/vendor/symfony*, se guarda el propio *framework* Symfony. Por lo tanto, no es necesario instalar Symfony de manera independiente ya que el sistema tiene la versión 1.4 autocontenida. La ruta donde se especifica la ruta donde se encuentra el *framework* Symfony, está descrita en el archivo *config/ProjectConfiguration.class.php*. Si abrimos este fichero nos encontraremos con una línea similar a esta:

```
require_once dirname(__FILE__).
'/../lib/vendor/symfony/lib/autoload/sfCoreAutoload.class.php';
```

Esta es la línea que especifica la ruta de Symfony. Es importante asegurarse que en esa ruta realmente se encuentra el *framework*.

Después de descomprimir el producto, se debe modificar los permisos de las carpetas cache, log para ello se ejecuta el siguiente comando:

```
$ chmod 777 cache/
$ chmod 777 log/
```

A.4. Creación de la base de datos

En esta sección, crearemos en primer lugar la base de datos, que utilizará la aplicación web. Esta base de datos, se creará mediante la ejecución de un script SQL y el nombre de la misma será «bd5». Posteriormente, se configurará Symfony para que sea accesible a la base de datos recién creada.

A.4.1. Creación del modelo de datos

Otra de las opciones para crear el modelo de datos, es ejecutar los scripts *crea_bd.sql* e *inserta_db.sql* que se encuentran en la carpeta *doc/scripts/*. Esta carpeta se encuentra dentro del archivo comprimido que se facilita. El primero de los scripts crea las tablas del modelo relacional, mientras que el segundo scripts inserta los datos en las tablas ya creadas.

Para ejecutar los scripts mencionados anteriormente se debe ejecutar los siguientes comandos:

```
$ mysql -u user -ppasword < crea_bd.sql
$ mysql -u user -ppasword < insertar_db.sql
```

Es importante mencionar que los scripts se han probando y ejecutado correctamente en el sistema gestor de bases de datos MySQL. Aunque el lenguaje SQL, suele ser muy parecido entre sistema gestores de base de datos puede que haya algún tipo de diferencia léxica.

Una vez acabado estos pasos correctamente, se puede asegurar que la base de datos ya está lista para ser utilizada. Por tanto, sólo queda la configuración de Symfony para que pueda acceder a la base de datos que se acaba de crear. Esta configuración se explica en el siguiente apartado. Es necesario recordar que el nombre de la base de datos recién creada es «bd5».

A.5. Configuración de la base de datos

Aunque Symfony tiene implementado un ORM *Object-Relational Mapping* y por tanto, es independiente de la base de datos utilizada, al menos se debe configurar unos parámetros mínimos para que el *framework* acceda correctamente a la misma. Estos parámetros, se configuran en el archivo *databases.yml* que se encuentra en la carpeta *config/databases.yml* dentro de la raiz del proyecto. A continuación se muestra un ejemplo del fichero:

```
prod:
  doctrine:
    class: sfDoctrineDatabase
    param:
      dsn:      mysql:host=host_produccion;dbname=produccion
      username: -------
      password: -------
dev:
  doctrine:
    class: sfDoctrineDatabase
    param:
      dsn:      mysql:host=localhost;dbname=desarrollo
      username: -------
      password: -------
```

A través de las etiquetas dev, prod, test se pueden definir diferentes configuraciones para los entornos development, production y test. Después de especificar la etiqueta de entorno, se añade la etiqueta que da información del ORM y de la clase (etiqueta class) que utilizará el entorno, en nuestro caso siempre es Doctrine aunque Symfony también soporta el ORM Propel. En el siguiente listado se detalla las diferentes opciones que tiene la etiqueta «param»:

- phptype. Especifica el fabricante de la base de datos.
- hostspec. Especifica el host.
- database. Especifica el nombre de la base de datos.

- username. Usuario de acceso a la base de datos.
- password. Contraseña de acceso a la base de datos.
- port. Puerto de escucha.
- encoding. Codificación utilizada para crear la tabla.
- persistent. Especifica si utilizar conexiones persistentes.
- dns. Se corresponde con las siglas *data source name*, y es una forma abreviada de detallar la información sobre el tipo de la base de datos, host y nombre de la misma.

A.6. Creación de usuarios

La aplicación tiene una sección mediante la cual se puede crear usuarios, pero sólo si un usuario con permisos de administrador ha accedido al sistema. Por este motivo, la ejecución anterior del script *inserta_db.sql* crea un usuario administrador de prueba con nombre de usuario «admin» y password «admin», para que el administrador pueda acceder a la sección de creación de usuarios mencionada anteriormente.

También es posible la creación de usuarios mediante comandos Symfony, desde la consola del sistema. Para la creación de un usuario se debe ejecutar el siguiente comando:

```
php symfony guard:create-user email nombre password
```

Para que cierto usuario tenga permisos de administrador, se debe ejecutar el siguiente comando:

```
php symfony guard:promote nombre_usuario
```

Con el comando anterior, el usuario con el nombre «nombre_usuario» pasará a ser administrador.

A.7. Configuración del servidor web

En esta sección se explicará cómo configurar el servidor web Apache para que el proyecto sea accesible desde la web.

En primer lugar, hay que localizar el archivo *httpd.conf*, dependiendo de la distribución Linux que se tenga instalada estará en una ubicación o en otra. Lo más normal, es que el archivo se encuentre en la ruta */etc/apache2/httpd.conf*.

Una vez abierto el fichero, se debe insertar la siguiente información al final del fichero:

```
# Be sure to only have this line once in your configuration
NameVirtualHost 127.0.0.1:8080

# This is the configuration for your project
Listen 127.0.0.1:8080

<VirtualHost 127.0.0.1:8080>
  DocumentRoot "sfproject/web"
  DirectoryIndex index.php
  <Directory "sfproject/web">
    AllowOverride All
    Allow from All
  </Directory>

  Alias /sf sfproject/lib/vendor/symfony/data/web/sf
  <Directory "sfproject/lib/vendor/symfony/data/web/sf">
    AllowOverride All
    Allow from All
  </Directory>
</VirtualHost>
```

Importante. Para que la configuración funcione, se tiene que reemplazar la palabra *sfproject* por la ruta donde se ha descomprimido el producto.

El alias */sf* permite dar acceso a las imágenes y ficheros javascript necesarios para que se muestre de forma apropiada la barra de depuración y las páginas por defecto de Symfony. Aunque en esta aplicación no se utilizará

la barra de depuración, ya que es un sistema probado, no está de más tener configurado esta característica de Symfony.

La configuración realizada anteriormente hace que Apache escuche por el puerto 8080 de la máquina, así que la aplicación será accesible a través de la siguiente URL:

```
http://localhost:8080/
```

Se puede cambiar el puerto de escucha por cualquier otro, pero es mejor los puertos superiores a 1024 ya que no necesitan derechos de administración.

Apéndice B

Manual de usuario

.1. Introducción

Este manual ha sido elaborado con la intención de ofrecer la información necesaria para utilizar todas las características que la aplicación ofrece. El manual sigue una estructura concreta, empezando por mostrar la estructura general de la aplicación y siguiendo con las operaciones que la aplicación puede realizar.

Esta aplicación tiene un control de usuarios, y por tanto el aspecto de la aplicación web varía dependiendo de las credenciales del usuario autentificado. Existen tres tipos de usuarios:

- Usuario anónimo
- Lingüista
- Administrador

.2. Estructura general de la aplicación

En este apartado se analiza la estructura común que forman todas las páginas de la aplicación, véase Figura 1. Cuando se trabaja diseñando aplicaciones web para usuarios de diferentes rasgos informáticos, es muy importante guardar una estructura fija en toda la aplicación. De esta manera, los usuarios tendrán una visión más clara de dónde se encuentra cierta información.

La estructura principal de la página web se divide en tres partes: cabecera, contenido, pie. Cada una de estas partes se dividen a su vez en varias cajas o subpartes. En los siguientes apartados, se profundiza en cada una de estas partes.

.2.1. Partes de la cabecera

Las partes en las que se divide la cabecera son: barra de usuario, logotipo y título de la aplicación. En la barra de usuario aparece el formulario necesario para la autentificación de usuarios. Cuando un usuario está ya autentificado, muestra el alias o nombre del usuario en la aplicación. En la caja destinada al logo, aparece el logo elegido para identificar a la aplicación web. Por último, se encuentra la caja de título. En dicha caja, aparece el nombre que recibe la aplicación.

.2.2. Partes del contenido

Esta parte es la que más espacio de página ocupa. Se divide en las siguientes cajas o subpartes: barra de navegación, panel lateral izquierdo y título del

contenido. La barra de navegación sirve de ayuda para navegar por todas las páginas de la aplicación web. Es un elemento que siempre está visible y en ningún momento se oculta. Este elemento, como ya se ha explicado anteriormente, puede variar dependiendo del usuario autentificado. El panel lateral izquierdo muestra la ayuda de la aplicación. En todas las páginas, tendrá dos enlaces: Manual de usuario, Manual de sección. El título del contenido, informa sobre el tipo de contenido de la página actual.

.2.3. Partes del pie de página

Esta parte se usa solamente como elemento estético, siendo una separación entre la parte del contenido y el final de la página.

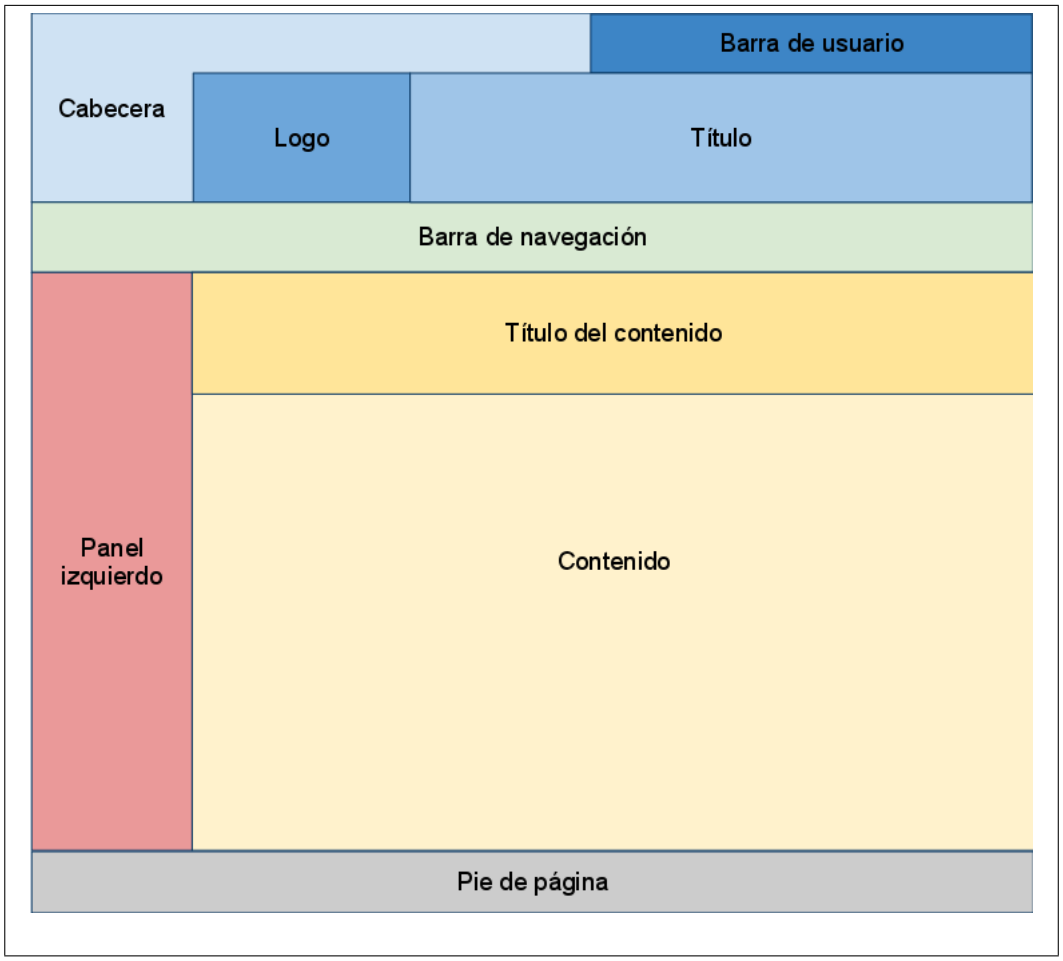


Figura 1: *Estructura de la página web*

.3. Operaciones

En este apartado se explicarán todas las operaciones que la aplicación puede realizar. Todas ellas se pueden acceder desde el menú o barra de navegación de la aplicación.

.3.1. Iniciar consulta

Esta operación puede ser realizada por cualquier usuario que haya accedido a la página web. A través de esta operación, se puede realizar consultas a la base de datos de la aplicación. En primer lugar, la página contiene una introducción y explicación de esta sección de la aplicación. Después de esta introducción, aparece un formulario donde el usuario elige aquello que desea consultar. Una vez elegida la información a consultar, se debe pulsar el botón Aceptar. Cuando se pulsa en dicho botón, la base de datos realizará la consulta pedida. Dependiendo de la información elegida en el formulario, la consulta tardará más o menos tiempo. El formulario donde se elige la información a consultar, es dinámico, es decir, cuando se pulsa en ciertos elementos otros se desactivan. De esta manera, se asegura que la consulta que el usuario realiza tenga sentido dentro del ámbito lingüístico. Una vez realizada la consulta por parte de la base de datos, el resultado se muestra en una tabla, véase Figura 2. Esta tabla se puede ordenar pulsando el campo por el que se quiere ordenar, véase Figura 2 y también, se puede exportar a un archivo tipo Excel o PDF. Para ello, basta con pulsar en los iconos que se muestran encima de la cabecera de la tabla, véase Figura 3.

Existen consultas, que exceden el límite de filas que una tabla HTML puede soportar para que la navegación web sea fluida, véase Figura 4. Para estos casos, se puede mostrar únicamente un rango de filas de la tabla total. Para ello, aparecerá un pequeño formulario donde se debe introducir la fila de inicio y la fila final del rango. Una vez introducido, estos datos pulsando el botón Mostrar, aparecerá la tabla con el rango de filas deseado. Véase la Figura 5.

Otra de las posibilidades es recorrer la tabla por páginas. Para ello, se debe ir pulsando en el enlace «Siguiente página» que aparece en la parte derecha de la tabla.

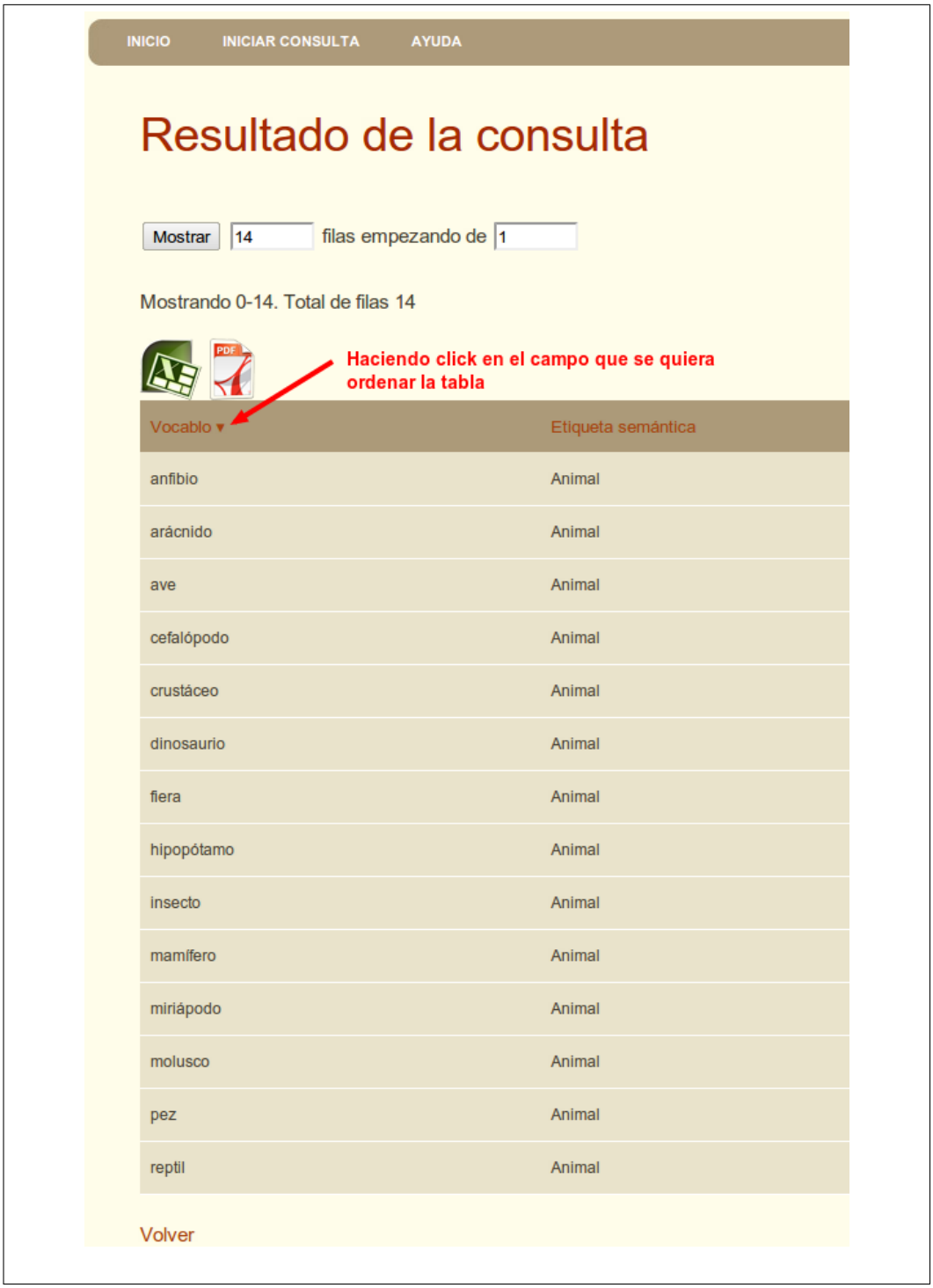


Figura 2: *Detalle de la tabla de salida tras ejecutar una consulta*

Figura 3: *Detalle de cómo exportar la tabla de salida a otros formatos*

Figura 4: *Detalle del formulario para mostrar el rango de filas deseado*

Figura 5: *Tabla de salida después de modificar el rango de filas*

.3.2. Insertar datos

Esta operación sólo puede ser realizada por personas con el rol lingüista o administrador. Mediante esta operación se inserta nuevos datos en la base de datos de la aplicación. En primer lugar, se muestra una introducción y explicación de las acciones que se pueden realizar. Después se muestra un listado con los conceptos lingüísticos que se pueden insertar, véase Figura 6.

Figura 6: *Inicio de la sección de «Inserción de datos»*

Una vez elegido uno de ellos, se muestra un formulario con los datos necesarios que se pueden insertar, relacionado con el concepto elegido. Estos tipos de formularios, son dinámicos, es decir, dependiendo de las selecciones que se hayan realizado en los campos, aparecerán o desaparecerán elementos del formulario. Pulsando en el botón Insertar datos se introducirán los datos especificados por el usuario. Importante: Asegurarse de que los datos introducidos sean correctos, ya que no es posible dar marcha atrás.

Cuando se hayan insertado los datos correctamente, se muestra un mensaje informando de que la inserción se ha realizado sin problemas. A continuación se explicará cada uno de los formularios que componen esta sección.

.3.3. Insertar etiqueta semántica

Mediante este formulario, véase Figura 7, se puede insertar una etiqueta semántica en la base de datos de la aplicación. A continuación, se explica los elementos que componen el formulario:

- Expresión. En este campo se inserta un texto que se corresponde con el nombre de la propia etiqueta semántica
- Inferencia semántica.
- Es sinónima. Si se activa esta caja de validación, se debe seleccionar una etiqueta sinónima en el desplegable del elemento «Sinónima a» del formulario.
- Sinónima a. Este elemento es un desplegable que sólo se activará si se ha seleccionado el elemento «Es sinónima» del formulario.
- Clasificada. Es una caja de validación. Si se selecciona, se activará el elemento «Clasificada dentro de».
- Clasificada dentro de. Es un desplegable que sólo se activará en el caso en que se haya seleccionado el elemento «Clasificada» del formulario. De esta manera se podrá insertar la clasificación de la etiqueta semántica.
- Supone (sin valor) función léxica. Es una caja de validación. Si se selecciona, se activará el elemento «Función léxica (sin valor)» del formulario.
- Función léxica (sin valor). Es un desplegable que sólo aparecerá activado cuando se haya seleccionado el elemento «Supone (sin valor)

función léxica». Mediante este elemento, se puede insertar la función léxica asociada a la etiqueta semántica que se quiere insertar. Se puede dar el caso de que no exista la función léxica en el sistema. Para estos casos, la aplicación ofrece la oportunidad de insertar la función léxica en este mismo formulario. Para ello, debe seleccionar en el desplegable la opción «Otros». Cuando se haya seleccionado, aparecerá una copia del formulario de inserción de función léxica. Para obtener información de este formulario, véase la sección «Insertar función léxica»

- Observaciones. Es un campo de texto que permite introducir algunas observaciones sobre la inserción de la etiqueta semántica. Cuando el usuario esté seguro de haber rellenado correctamente el formulario, se debe pulsar al botón «Insertar» para que los datos introducidos tengan efecto sobre la aplicación.

Figura 7: *Detalle del formulario de inserción de etiqueta semántica*

.3.4. Insertar función léxica

Mediante este formulario, véase Figura 8, el usuario acreditado podrá insertar una función léxica en la base de datos de la aplicación. A continuación, se explica los elementos que componen el formulario:

- Expresión. En este campo se inserta un texto que se corresponde con el nombre de la función léxica que se desea insertar.
- Núcleo. En este campo se inserta el núcleo de la función léxica
- Superíndice. En este campo se inserta el superíndice de la función léxica
- Subíndice. En este campo se inserta el subíndice de la función léxica
- Estándar. Es un campo numérico en el que sólo se debe introducir un carácter numérico.
- Clase función léxica. Este campo es un desplegable donde se elige aquella función léxica que forma clase de la función léxica a introducir. Si no existe la clase función léxica deseada, se puede introducir una nueva buscando el valor «Añadir nueva» en el propio desplegable. Al seleccionar dicho valor aparecerá un nuevo formulario donde se puede añadir una nueva clase de función léxica.
- Es combinación. Este campo es una caja de validación. Si se selecciona, se activan dos desplegables para seleccionar las combinaciones de funciones léxicas que se quieran seleccionar.
- Combinación1 de. Este desplegable sólo se activará si se ha seleccionado el campo «Es combinación». En este campo, se elige la primera combinación de la función léxica que se va a insertar.

- Combinación2 de. Este desplegable sólo se activará si se ha seleccionado el campo «Es combinación». En este campo, se elige la segunda combinación de la función léxica que se va a insertar.

- Descripción semántica. Es un campo de texto para introducir la descripción semántica de la función léxica. No es un campo obligatorio, por tanto puede estar vacío.

- Añadir glosa existente. Este campo es una caja de validación. Si se selecciona, se activará el campo «Glosa» del formulario. Por otro lado, si se quita la selección, el campo «Nueva glosa» pasará a estar activo. Este campo debe ser seleccionado en el caso en que se quiera añadir una glosa relacionada con la función léxica que se va a insertar.

- Glosa. Este campo es un desplegables que sólo se activa si se ha seleccionado el campo «Añadir glosa existente».

- Nueva glosa. Este campo es una caja de texto donde se puede insertar una nueva glosa asociada a la función léxica que se desea insertar. Este campo aparecerá inactivo en el caso en que se haya seleccionado el campo «Añadir glosa existente».

- Añadir paráfrasis existente. Este campo es una caja de validación. Si se selecciona, se activará el campo «Paráfrasis» del formulario. Por otro lado, si se quita la selección, el campo «Nueva paráfrasis» pasará a estar activo. Este campo debe ser seleccionado en el caso en que se quiera añadir una paráfrasis relacionada con la función léxica que se va a insertar.

- Paráfrasis. Este campo es un desplegables que sólo se activa si se ha seleccionado el campo «Añadir paráfrasis existente».

- Nueva paráfrasis. Este campo es una caja de texto donde se puede insertar una nueva paráfrasis asociada a la función léxica que se desea insertar. Este campo aparecerá inactivo en el caso en que se haya seleccionado el campo «Añadir paráfrasis existente».

- Observaciones. Es un campo de texto que permite introducir algunas observaciones sobre la inserción de la función léxica. Cuando el usuario esté seguro de haber rellenado correctamente el formulario, se debe pulsar al botón «Insertar» para que los datos introducidos tengan efecto sobre la aplicación.

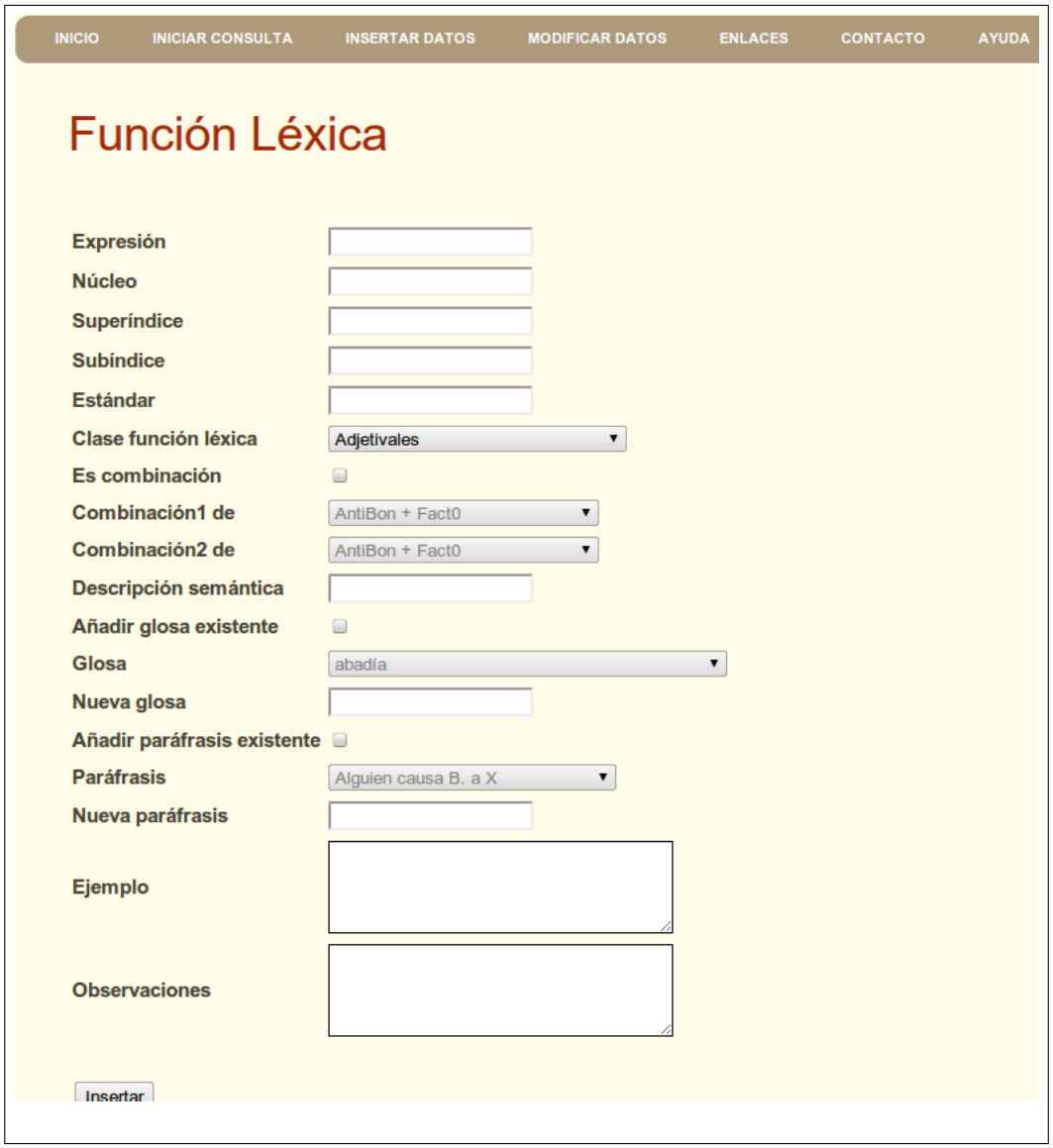


Figura 8: *Detalle del formulario de inserción de función léxica*

.3.5. Insertar paráfrasis

Mediante este formulario, véase Figura 9, el usuario acreditado podrá insertar una paráfrasis en la base de datos de la aplicación. A continuación, se explica los elementos que componen el formulario:

- Expresión. En este campo se inserta un texto que se corresponde con el nombre de la paráfrasis que se desea insertar.

- Función léxica. Este campo es un desplegable donde se selecciona la función léxica asociada a la paráfrasis que se desea insertar. Es posible insertar una nueva función léxica si no se encuentran en la base de datos de la aplicación. Para ello, se debe seleccionar en este campo el valor «Otros». Una vez seleccionado, aparecerá un nuevo formulario para insertar una nueva función léxica. las funciones que

- Es sinónima. Este campo es una caja de validación. Se debe seleccionar cuando se quiere insertar una etiqueta semántica sinónima a la paráfrasis que se desea insertar. Cuando se selecciona este campo, se activará el campo «Sinónima a»para elegir dicha etiqueta semántica.

- Sinónima a. Este campo es un desplegable donde se elige la etiqueta semántica sinónima a la paráfrasis que se está insertando. Sólo aparecerá activo cuando se haya seleccionado el campo «Es sinónima».

- Observaciones. Es un campo de texto que permite introducir algunas observaciones sobre la inserción de la paráfrasis. Cuando el usuario esté seguro de haber rellenado correctamente el formulario, se debe pulsar al botón «Insertar» para que los datos introducidos tengan efecto sobre la aplicación.

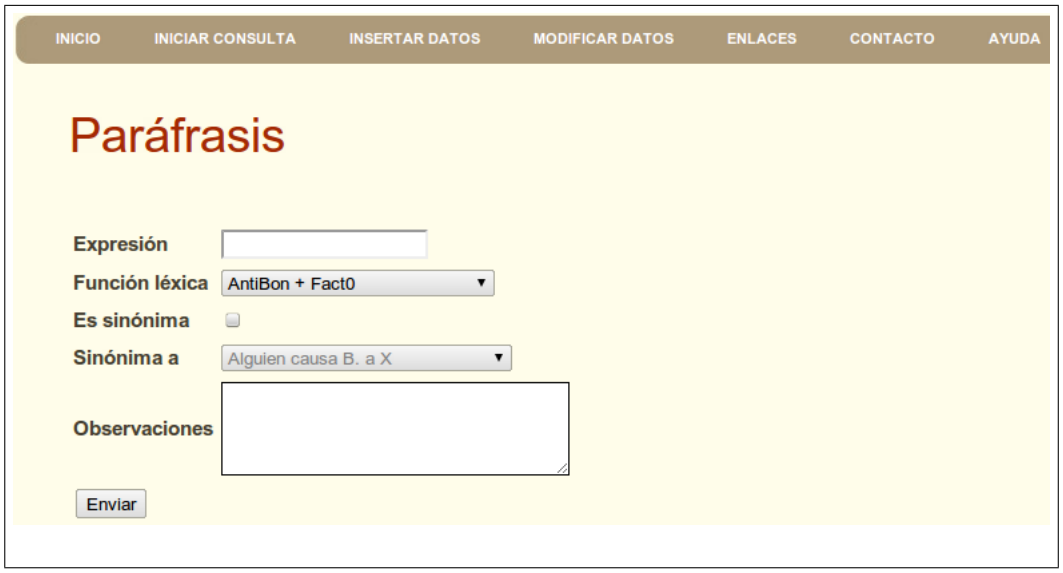


Figura 9: *Detalle del formulario de inserción de paráfrasis*

.3.6. Insertar concepto

Mediante este formulario, véase Figura 10, el usuario acreditado podrá insertar un concepto en la base de datos de la aplicación. A continuación, se explica los elementos que componen el formulario:

- Expresión. En este campo se inserta un texto que se corresponde con el nombre del concepto que se desea insertar.
- Equivalente a etiqueta semántica. Este campo es una caja de validación. Se debe seleccionar cuando se quiere insertar una etiqueta semántica equivalente al concepto que se desea insertar. Cuando se selecciona este campo, se activará el campo «Etiqueta semántica»para elegir dicha etiqueta semántica.
- Etiqueta semántica. Este campo es un desplegable donde se selecciona

la etiqueta semántica equivalente al concepto que se desea insertar. Es posible insertar una nueva etiqueta semántica si no se encuentran en la base de datos de la aplicación. Para ello, se debe seleccionar en este campo el valor «Otros». Una vez seleccionado, aparecerá un nuevo formulario para insertar una nueva etiqueta semántica.

- Lema. Este campo es un desplegable donde se selecciona el lema del actante. Es posible insertar un nuevo lema si no se encuentran en la base de datos de la aplicación. Para ello, se debe seleccionar en este campo el valor «Otros». Una vez seleccionado, aparecerá un nuevo formulario para insertar un nuevo lema.

- Clasificado en unidad léxica. Este campo es una caja de validación. Se debe seleccionar en el caso en que se quiera clasificar el concepto a insertar, con una unidad léxica. Cuando se selecciona este campo, se activará el campo «Unidad Léxica»para elegir dicha unidad léxica.

- Unidad léxica. Este campo es un desplegable donde se selecciona la unidad léxica con la que se quiere clasificar el concepto a insertar. Es posible insertar una nueva unidad léxica, si no se encuentra en la base de datos de la aplicación. Para ello, debe seleccionar en este desplegable el valor «Otros». Una vez seleccionado, aparecerá un nuevo formulario para insertar una nueva unidad léxica.

- Observaciones. Es un campo de texto que permite introducir algunas observaciones sobre la inserción del concepto. Cuando el usuario esté seguro de haber rellenado correctamente el formulario, se debe pulsar al botón «Insertar»para que los datos introducidos tengan efecto sobre la aplicación.

Figura 10: *Detalle del formulario de inserción de concepto*

.3.7. Insertar unidad léxica

Mediante este formulario, véase Figura 11, el usuario acreditado podrá insertar una unidad léxica en la base de datos de la aplicación. A continuación, se explica los elementos que componen el formulario:

- Vocablo acepción. En este campo se inserta un texto que se corresponde con el nombre de la unidad léxica que se desea insertar.
- Discriminante. Preguntar qué es discriminante?
- Glosa expresa función léxica. Este campo es un caja de validación, que se debe seleccionar en el caso en que la unidad léxica sea a su vez una glosa que expresa cierta función léxica. Cuando se selecciona este

campo, se activará el campo «Función Léxica»para elegir dicha función léxica.

- Función léxica. Este campo es un desplegable donde se selecciona la función léxica que expresa la glosa. Es posible insertar un nueva función léxica si no se encuentran en la base de datos de la aplicación. Para ello, se debe seleccionar en este campo el valor «Otros». Una vez seleccionado, aparecerá un nuevo formulario para insertar una nueva función léxica.

- Observaciones. Es un campo de texto que permite introducir algunas observaciones sobre la inserción de la unidad léxica.

- Clasificado en concepto. Este campo es una caja de validación. Se debe seleccionar en el caso en que se quiera clasificar la unidad léxica a insertar, con un concepto. Cuando se selecciona este campo, se activará el campo «Concepto»para elegir dicho concepto.

- Concepto. Este campo es un desplegable donde se selecciona el concepto con el que se quiere clasificar la unidad léxica. Es posible insertar un nuevo concepto, si no se encuentra en la base de datos de la aplicación. Para ello, debe seleccionar en este desplegable el valor «Otros». Una vez seleccionado, aparecerá un nuevo formulario para insertar un nuevo concepto.

- Actante de lema. Este campo es una caja de validación. Se debe seleccionar en el caso en que la unidad léxica a insertar, sea un actante de un lema. Cuando se selecciona este campo, se activará el campo «Lema»para elegir dicho lema.

- Lema. Este campo es un desplegable donde se selecciona el lema que será actante la unidad léxica. Es posible insertar un nuevo lema, si no se encuentra en la base de datos. Para ello, debe seleccionar en este desplegable el valor «Otros». Una vez seleccionado, aparecerá un nuevo formulario para insertar un nuevo lema.

- Es glosa de etiqueta semántica. Este campo es un caja de validación, que se debe seleccionar en el caso en que la unidad léxica sea a su vez una glosa que expresa cierta etiqueta semántica. Cuando se selecciona este campo, se activará el campo «Etiqueta semántica»para elegir dicha etiqueta semántica.

- Etiqueta semántica. Este campo es un desplegable donde se selecciona la etiqueta semántica que expresa la glosa. Es posible insertar un nueva etiqueta semántica si no se encuentran en la base de datos de la aplicación. Para ello, se debe seleccionar en este campo el valor «Otros». Una vez seleccionado, aparecerá un nuevo formulario para insertar una nueva etiqueta semántica. Cuando el usuario esté seguro de haber rellenado correctamente el formulario, se debe pulsar al botón «Insertar»para que los datos introducidos tengan efecto sobre la aplicación.

Figura 11: *Detalle del formulario de inserción de unidad léxica*

.3.8. Insertar lema

En esta sección se puede insertar un nuevo lema en la base de datos de la aplicación. Se compone de varios formularios ya que previamente, se debe escribir el vocablo del lema que se quiere insertar, véase Figura 12.

Después de escribir el lema que se quiere insertar, la aplicación buscará si existe el lema insertado. En caso de que no exista, se mostrará un formulario con varios campos, véase Figura 13, para insertar el lema. En caso contrario, se mostrará una lista de acepciones asociadas al vocablo insertado anteriormente. En la misma página, se pregunta si se

quiere añadir una nueva acepción. Si el usuario selecciona esta opción, se mostrará un formulario para insertar dicha acepción, véase Figura 14.

Figura 12: *Detalle del formulario principal de inserción de lema*

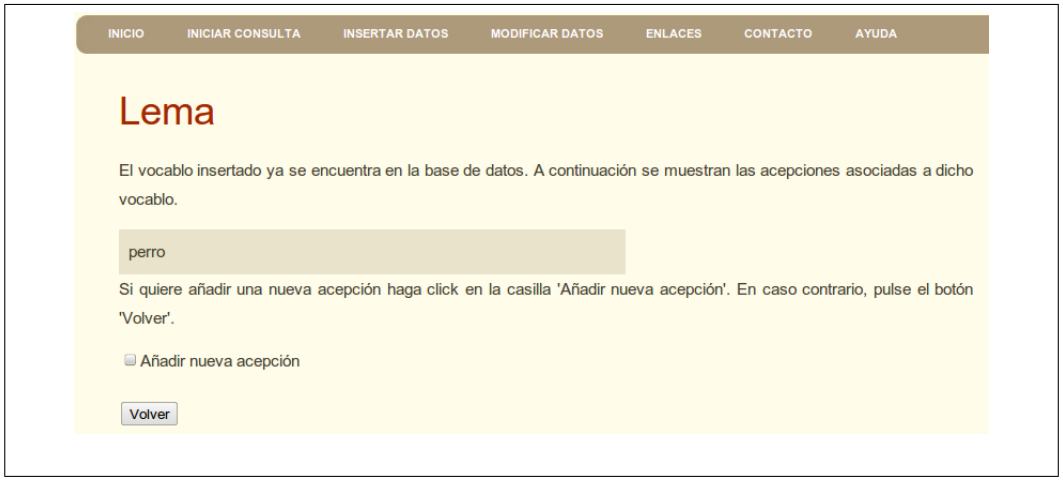


Figura 13: *Detalle del formulario de inserción de lema tras insertar el lema a modificar*

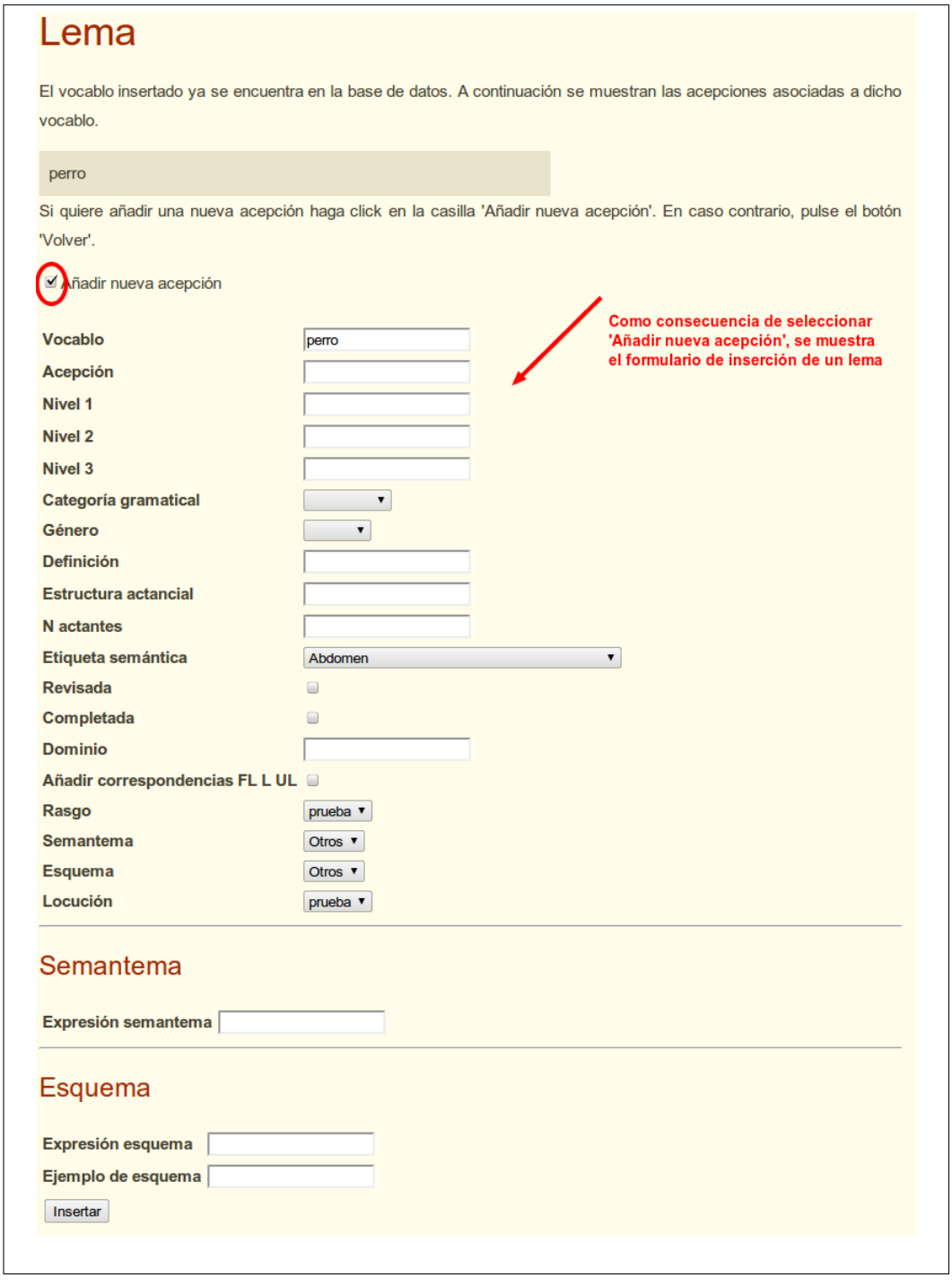


Figura 14: *Detalle del formulario de inserción de una acepción a un lema*

Como se ha podido observar en la segunda imágen, se activa un formu-

lario para insertar un nuevo lema. Este formulario es idéntico al que se muestra cuando el vocablo del lema insertado no existe en la base de datos de la aplicación. Véase primer caso de esta misma sección. Por este motivo sólo se explicará una vez este formulario:

- Nivel1.
- Nivel2.
- Nivel3.
- Categoría gramatical. Es un campo desplegable donde se elige la categoría gramatical del lema que se va a insertar. En el campo desplegable aparece la clasificación típica de categorías gramaticales, por tanto no se da la posibilidad de añadir ninguna más.
- Género. Es un campo desplegable que permite seleccionar el género del que se va a introducir en el sistema.
- Definición. Es un campo de texto opcional que permite introducir una pequeña definición del lema que se va a introducir en el sistema. Como máximo se permite introducir 250 caracteres.
- Estructura actancial. Es un campo de texto opcional que permite especificar la estructura actancial del lema. El número de caracteres máximos de este campo es 100.
- Revisada. Este campo es una caja de validación. La selección de este campo indica que el lema que se va a introducir se ha revisado, y por tanto es correcto.
- Completada.

- Dominio. Es un campo de texto opcional donde se introduce el dominio del lema. El número de máximo de caracteres de este campo es 75. Añadir correspondencias FL L UL. Es una caja de validación. Cuando se selecciona, muestra un mensaje advirtiendo que cuando se pulse el botón «Insertar», se mostrará automáticamente un formulario donde se podrá insertar los valores de la correspondencia función léxica y lema.

- Rasgo. Este campo es un desplegable que permite seleccionar el rasgo del lema que se va a introducir. Es posible añadir un nuevo rasgo. Para ello debemos seleccionar el valor «Otros»y automáticamente aparecerá un nuevo formulario.

- Semantema. Este campo es un desplegable que permite seleccionar el semantema del lema que se va a introducir. Es posible añadir un nuevo semantema. Para ello debemos seleccionar el valor «Otros»y automáticamente aparecerá un nuevo formulario.

- Esquema. Este campo es un desplegable que permite seleccionar el esquema del lema que se va a introducir. Es posible añadir un nuevo esquema. Para ello debemos seleccionar el valor «Otros»y automáticamente aparecerá un nuevo formulario.

- Locución. Este campo es un desplegable que permite seleccionar la locución del lema que se va a introducir. Es posible añadir una nueva locución. Para ello debemos seleccionar el valor «Otros»y automáticamente aparecerá un nuevo formulario.

- Expresión rasgo. Este campo es una caja de texto. Sólo aparecerá si se ha seleccionado el campo «Rasgo». Mediante este campo se permite introducir un nuevo rasgo.

- Expresión semantema. Este campo es una caja de texto. Sólo aparecerá si se ha seleccionado el campo «Semantema». Mediante este campo se permite introducir un nuevo semantema.

- Expresión esquema. Este campo es una caja de texto. Sólo aparecerá si se ha seleccionado el campo «Esquema». Mediante este campo se permite introducir un nuevo esquema.

- Expresión locución. Este campo es una caja de texto. Sólo aparecerá si se ha seleccionado el campo «locución». Mediante este campo se permite introducir una nueva locución.

Si el usuario ha seleccionado el campo «Añadir correspondencias FL L UL», aparece el siguiente formulario:

Este formulario es exactamente igual que aquel que se obtiene al pulsar el enlace «Insertar valores a función léxica y lema». Este enlace se encuentra en el inicio de la sección «Insertar datos». Cuando el usuario esté seguro de haber rellenado correctamente el formulario, se debe pulsar al botón «Insertar»para que los datos introducidos tengan efecto sobre la aplicación.

.3.9. Insertar clase función léxica

Mediante este formulario, véase Figura 15, el usuario acreditado podrá insertar una clase de función léxica en la base de datos de la aplicación. A continuación, se explica los elementos que componen el formulario:

- Expresión. En este campo se inserta un texto que se corresponde con el nombre de la clasificación de función léxica que se quiere insertar.

- Observaciones. Es un campo de texto que permite introducir algunas observaciones sobre la inserción de la clasificación de función léxica.

Cuando el usuario esté seguro de haber rellenado correctamente el formulario, se debe pulsar al botón «Insertar»para que los datos introducidos tengan efecto sobre la aplicación.

Figura 15: *Detalle del formulario de inserción de una clase de función léxica*

.3.10. Insertar valores a función léxica y lema

Este formulario permite insertar valores de unidad léxica sobre la correspondencia de función léxica y lema. El formulario presenta varias áreas diferentes, véase Figura 16. Explicación de las áreas del formulario:

- Área de función léxica. Este área presenta un formulario formado por un único desplegable, donde se debe elegir la función léxica deseada. Una vez elegida la función léxica, es posible obtener las correspondencias del mismo pulsando el botón Ver. En el momento en que se pulsa este botón, se modifica el área Correspondencias, mostrando todas las

correspondencias de la función léxica elegida. En el caso en que la función léxica elegida no tenga correspondencias, aparecerá un mensaje en el área de Correpondencias.

- Área de lema. Este área presenta un formulario formado por un único desplegable, donde se debe elegir el lema deseado. Una vez elegido el lema, es posible obtener las correspondencias del mismo pulsando el botón Ver. En el momento en que se pulsa este botón, se modifica el área Correspondencias, mostrando todas las correspondencias del lema elegido. De igual forma que pasaba en el área explicado anteriormente, si no existe correspondencias para el lema seleccionado se mostrará un mensaje informativo en el área Correspondencias.

- Área combinación. Este área permite realizar una consulta de las correspondencias existentes que resulten como combinación de los elementos seleccionados en las áreas: Función léxica y Lema. Por tanto, para realizar dicha consulta tan sólo hay que pulsar en el botón Combinar. A continuación, se modificará el área Correspondencias mostrando las posibles correspondencias de la combinación de los dos elementos explicados anteriormente. Si no existe correspondencia ninguna, se mostrará un mensaje informativo en el área Correspondencias.

- Área unidad léxica. Este área presenta un formulario formado por un formulario de selección múltiple. La diferencia de este formulario, con los anteriores es que permite seleccionar más de un elemento a la vez. Para realizar esta selección múltiple, basta con mantener pulsado el botón Ctrl del teclado mientras con el ratón se seleccionan los elementos deseados. La característica de selección múltiple se utiliza cuando se quiere insertar varios valores a la vez. Una vez elegida la unidad léxica,

es posible obtener las correspondencias del mismo pulsando el botón Ver. En el momento en que se pulsa este botón, se modifica el área Correspondencias, mostrando todas las correspondencias de la unidad léxica elegida. De igual forma que pasaba en el área explicado anteriormente, si no existe correspondencias para la unidad léxica seleccionada se mostrará un mensaje informativo en el área Correspondencias.

- Área de inserción de correspondencia. En este área se permite realizar la inserción de uno o varios valores de unidades léxicas. Como ya se ha explicado anteriormente, es posible insertar varios valores de unidades léxicas haciendo uso de la selección múltiple del área Unidad léxica. Una vez elegido los elementos función léxica, lema y unidad léxica deseados, se debe pulsar el botón Insertar para que la inserción tenga efecto.

- Área correspondencias. La única funcionalidad de este área es la de informar de las correspondencias de los elementos que el usuario desea. Sólo entra en funcionamiento cuando se pulsa los botones de: Ver, Combinar o Insertar.

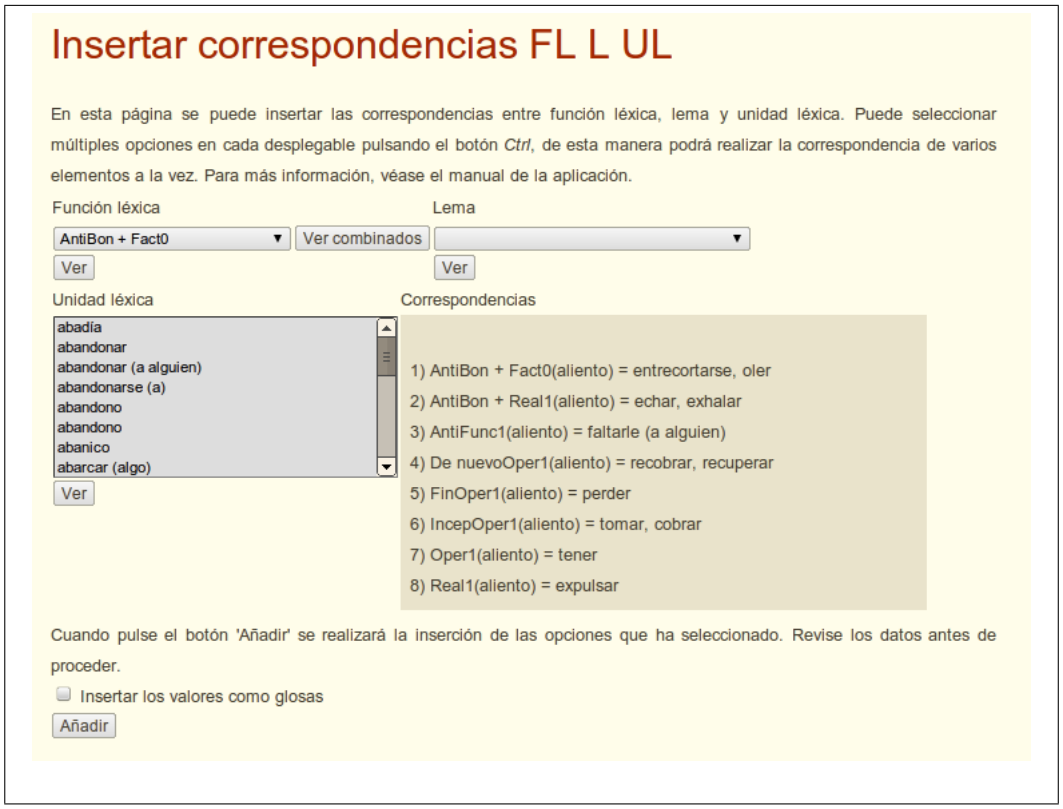


Figura 16: *Detalle del formulario de inserción de valores a función léxica y lema*

.3.11. Insertar valores a función léxica y etiqueta semántica

Este formulario permite insertar valores de unidad léxica sobre la correspondencia de función léxica y etiqueta semántica. El formulario presenta varias áreas diferentes, véase Figura 17. Explicación de las áreas del formulario:

- Área de función léxica. Este área presenta un formulario formado por un único desplegable, donde se debe elegir la función léxica deseada. Una vez elegida la función léxica, es posible obtener las correspondencias del mismo pulsando el botón Ver. En el momento

en que se pulsa este botón, se modifica el área Correspondencias, mostrando todas las correspondencias de la función léxica elegida. En el caso en que la función léxica elegida no tenga correspondencias, aparecerá un mensaje en el área de Correpondencias.

- Área de etiqueta semántica. Este área presenta un formulario formado por un único desplegable, donde se debe elegir la etiqueta semántica deseada. Una vez elegido dicha etiqueta semántica, es posible obtener las correspondencias de la misma pulsando el botón Ver. En el momento en que se pulsa este botón, se modifica el área Correspondencias, mostrando todas las correspondencias de la etiqueta semántica elegida. De igual forma que pasaba en el área explicado anteriormente, si no existe correspondencias para la etiqueta semántica seleccionada se mostrará un mensaje informativo en el área Correspondencias.

- Área combinación. Este área permite realizar una consulta de las correspondencias existentes que resulten como combinación de los elementos seleccionados en las áreas: Función léxica y Etiqueta semántica. Por tanto, para realizar dicha consulta tan sólo hay que pulsar en el botón Combinar. A continuación, se modificará el área Correspondencias mostrando las posibles correspondencias de la combinación de los dos elementos explicados anteriormente. Si no existe correspondencia ninguna, se mostrará un mensaje informativo en el área Correspondencias.

- Área unidad léxica. Este área presenta un formulario formado por un formulario de selección múltiple. La diferencia de este formulario, con los anteriores es que permite seleccionar más de un elemento a la vez. Para realizar esta selección múltiple, basta con

mantener pulsado el botón Ctrl del teclado mientras con el ratón se seleccionan los elementos deseados. La característica de selección múltiple se utiliza cuando se quiere insertar varios valores a la vez.

- Una vez elegida la unidad léxica, es posible obtener las correspondencias del mismo pulsando el botón Ver. En el momento en que se pulsa este botón, se modifica el área Correspondencias, mostrando todas las correspondencias de la unidad léxica elegida. De igual forma que pasaba en el área explicado anteriormente, si no existe correspondencias para la unidad léxica seleccionada se mostrará un mensaje informativo en el área Correspondencias.
- Área de inserción de correspondencia. En este área se permite realizar la inserción de uno o varios valores de unidades léxicas. Como ya se ha explicado anteriormente, es posible insertar varios valores de unidades léxicas haciendo uso de la selección múltiple del área Unidad léxica. Una vez elegido los elementos función léxica, etiqueta semántica y unidad léxica deseados, se debe pulsar el botón Insertar para que la inserción tenga efecto.
- Área correspondencias. La única funcionalidad de este área es la de informar de las correspondencias de los elementos que el usuario desea. Sólo entra en funcionamiento cuando se pulsa los botones de: Ver, Combinar o Insertar.

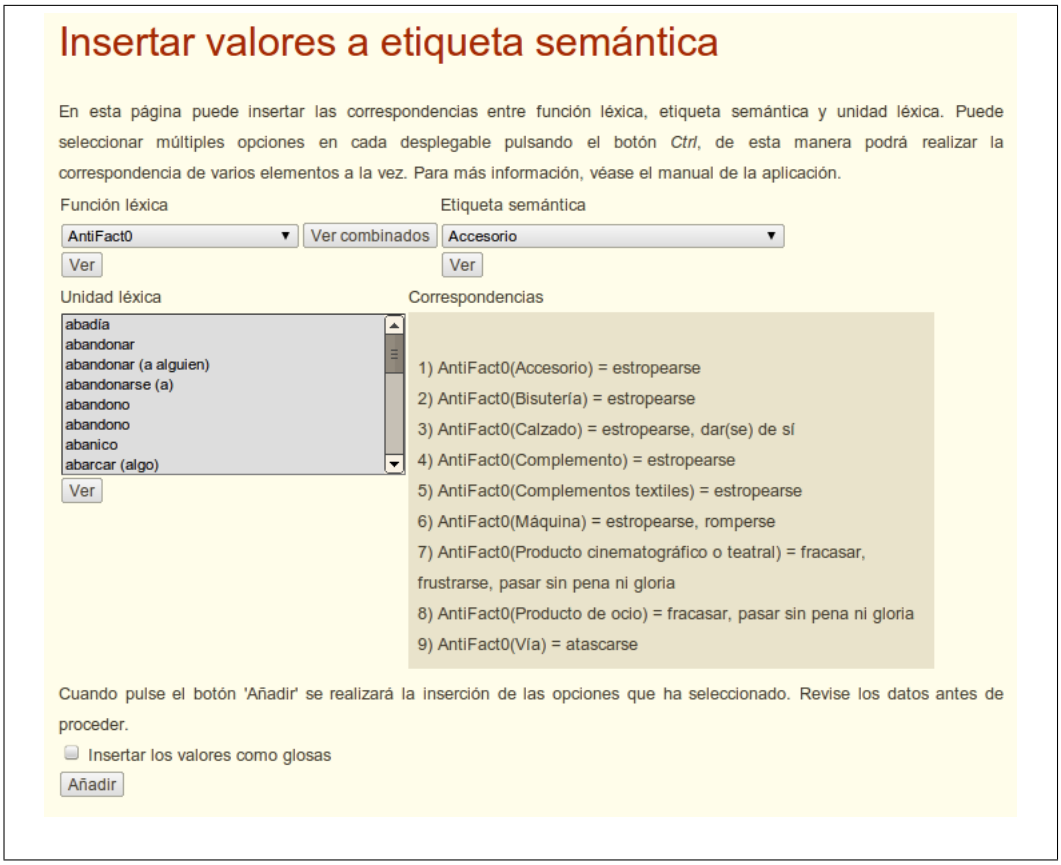


Figura 17: *Detalle del formulario de inserción de valores a función léxica y etiqueta semántica*

.3.12. Modificar datos

Esta operación sólo puede ser realizada por personas que tengan el rol de lingüista o administrador. Mediante esta operación se modifican o borran datos de la base de datos de la aplicación. En primer lugar, se muestra una introducción y explicación de las posibles acciones que se pueden realizar. Después se muestra un listado con los conceptos lingüísticos que se pueden modificar/borrar. Una vez elegido uno de ellos, se muestra un formulario con el elemento del concepto lingüístico elegido que se quiere modificar/borrar. Al pulsar Siguiente, aparecerá

otro formulario. Todos los formularios en esta sección se dividen en tres áreas, véase Figura 18.

Figura 18: *Ejemplo del formulario de modificación de datos de una etiqueta semántica*

Explicación de las diferentes áreas:

- Área de eliminación. En el área de eliminación se presenta una pregunta y un formulario de validación. En este formulario, se ofrece la posibilidad de eliminar el elemento seleccionado previamente, así como, todas sus relaciones. Cuando se pulsa en la casilla de validación, se puede observar que el resto de áreas se deshabilitan. El borrado del elemento y de todas sus dependencias, se realizará cuando se pulse el botón «Modificar».
- Área de modificación. En este área se pueden modificar las características del elemento seleccionado anteriormente. Las modi-

ficaciones se guardarán, en el momento en que se pulse el botón Modificar.

- Área de modificación de relaciones. Este área tiene en primer lugar un formulario con un desplegable dónde se encontrarán todas las posibles relaciones que el elemento puede tener. Al elegir una de estas relaciones y pulsar el botón Continuar, se mostrará bien un nuevo formulario en el caso de que haya relaciones para el elemento elegido, o bien, un texto en el que se informe que no existe ninguna relación para el elemento elegido. En el caso en que exista relaciones, el formulario que se muestra permite modificar dichas relaciones.

Cuando las modificaciones se han realizado con éxito, se mostrará un texto informativo explicando que todo ha ido correctamente.

Existen dos formularios que tienen ciertas diferencias con aquellos explicados anteriormente. Dichos formularios aparecen cuando en la pantalla principal de la sección de modificación de datos, se pulsa en Modificar valores de función léxica y lema o en Modificar valores de función léxica y etiqueta semántica. Una vez seleccionado en uno de ellos, se muestra un formulario que contiene dos desplegables. Estos desplegables se corresponden con dos conceptos lingüísticos que se deben elegir para poder realizar la modificación correctamente. Dichos conceptos que se deben elegir, son función léxica y lema, o bien, función léxica y etiqueta semántica, véase Figura 19. Cuando se haya elegido dichos elementos, se debe pulsar el botón «Siguiente».

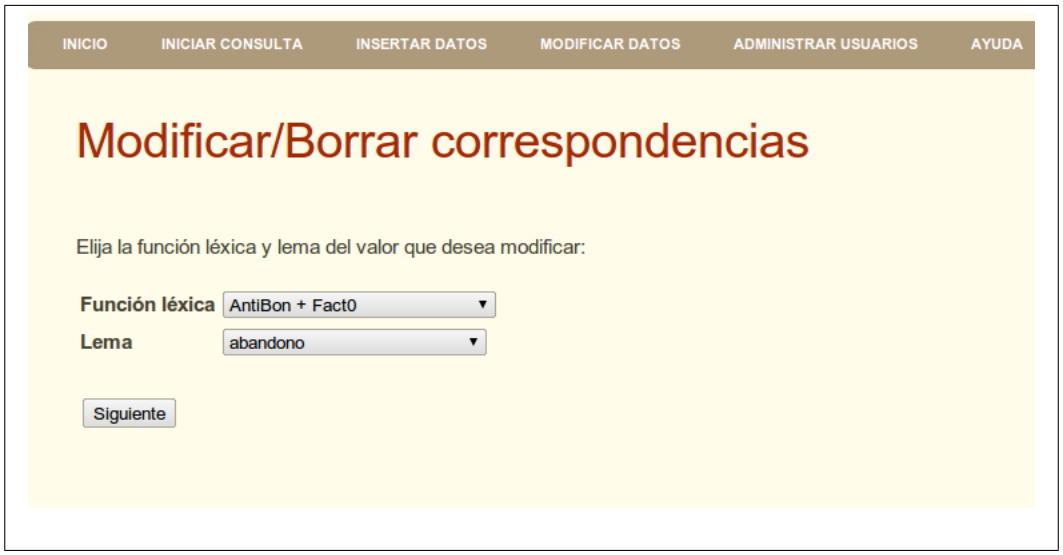


Figura 19: *Detalle del formulario de modificar correspondencias*

Tras pulsar el botón «Siguiente», aparece un nuevo formulario, véase Figura 20:

- Área de modificado de valor. En este área se puede modificar el valor de la unidad léxica asociado a los conceptos lingüísticos elegidos anteriormente. Se compone de dos desplegables, el primero de ellos se corresponde con el valor o valores de la unidad léxica en el momento de realizar la consulta. El otro desplegable, se correponde con todos los valores que una unidad léxica puede tener.

- Área de eliminación. En este área se puede eliminar completamente la correspondencia de los dos conceptos lingüísticos elegidos previamente y el valor de la unidad léxica, que aparece en el primer desplegable del área de modificación de valor. Este área, se compone solamente de una casilla de verificación. Cuando dicha casilla es pulsada, el segundo desplegable del área de modificación de valor se desactivará. En este

momento, pulsando el botón «Siguiente», se realizaría la eliminación de la correspondencia.

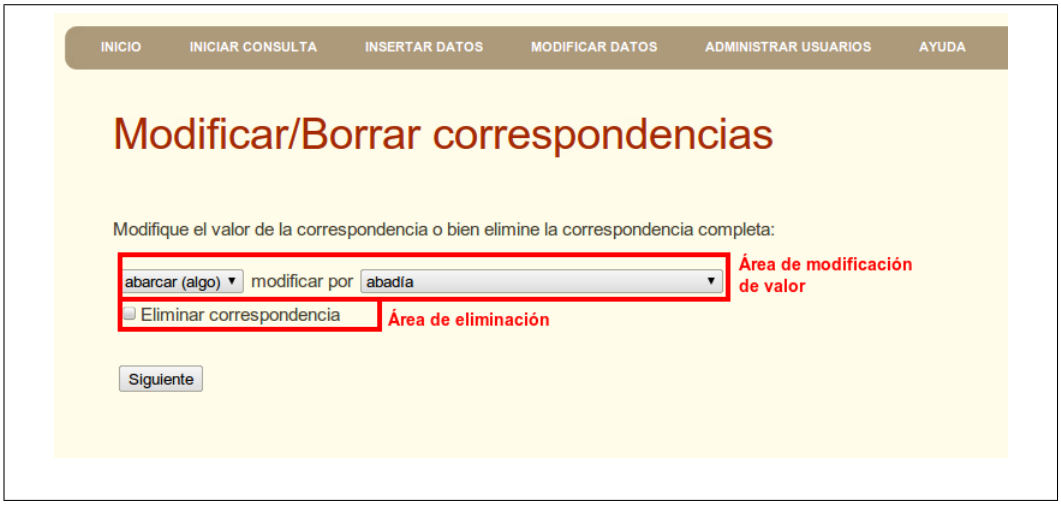


Figura 20: *Detalle del segundo formulario de modificar correspondencias*

Al igual que ocurría con el resto de formularios de la sección de modificación de datos, cuando los cambios se han realizado con éxito, se muestra un texto informativo explicando que todo ha ido correctamente.

.3.13. Administrar usuarios

En esta sección, se permite añadir, eliminar y mostrar los usuarios del sistema. Estas operaciones, sólo puede ser ejecutadas por usuarios que tengan permisos de administrador. Por tanto, sólo aparecerá el enlace a esta sección en aquellos usuarios que tengan dichos permisos. A continuación, se explica detalladamente dichas operaciones.

.3.13.1. Insertar usuario

Mediante esta operación la aplicación permite añadir un usuario al sistema. La inserción del usuario se hace mediante un sencillo formulario, véase Figura 21, que contiene los siguientes campos:

- Usuario. En este campo se debe introducir el nombre del usuario. Este nombre se tendrá que introducir, cuando el usuario se quiera autenticar en la aplicación. Importante: El nombre del usuario debe ser único en el sistema, por tanto si se introduce un nombre de usuario que ya está dado de alta en la aplicación, se producirá un error.
- Password. En este se debe introducir la contraseña del usuario, y al igual que en el campo anterior, es un dato que se deberá insertar en el momento de autenticarse en la aplicación
- Email. Mediante este campo se introduce el email de la persona, para tener guardado un contacto con el usuario que se va a insertar.Es importante recalcar que si un
- Administrador. Este campo es una caja de validación, seleccionarlo supone dar permisos de administrador al usuario que se va a insertar. Como se ha recordado en otras ocasiones, tener permisos de administrador supone tener acceso a todas las operaciones de la aplicación.

Todos los campos mencionados anteriormente son obligatorios.

Figura 21: *Detalle del formulario de inserción de usuario*

.3.13.2. Eliminar usuario

Esta operación permite eliminar usuarios del sistema. El borrado del usuario es una operación sencilla en la que tan sólo se tiene que elegir el usuario que se desee eliminar. Esta selección se realiza mediante un desplegable que muestra todos los usuarios del sistema. Véase Figura 22.

Figura 22: *Formulario para eliminar un usuario*

.3.13.3. Mostrar usuarios

Esta operación permite visualizar los usuarios del sistema. La aplicación mostrará los usuarios del sistema en tabla HTML con las columnas: id del usuario, nombre e email del mismo. Véase Figura 23.

Figura 23: *Tabla con los usuarios registrados en el sistema*

.3.13.4. Administrar base de datos

En esta sección se permite realizar y cargar copias de seguridad de la base de datos de la aplicación. Esta operaciones sólo pueden ser ejecutada por usuarios que tengan permisos de administrador. Por tanto, sólo aparecerá el enlace a esta sección en aquellos usuarios que tengan dichos permisos. A continuación, se explica detalladamente dichas operaciones.

.3.13.5. Realizar copia de seguridad

Mediante esta operación la aplicación realizará una copia de seguridad, que se volcará a un fichero con extensión SQL. EL usuario puede elegir que se guarde en el servidor de la aplicación o bien de forma local en su ordenador. Si

el usuario elige guardar la copia de seguridad en el servidor (opción servidor del desplegable), entonces debe facilitar una ubicación dentro del servidor que tenga permisos de escritura, en caso contrario, la copia de seguridad no se guardará. Si el usuario prefiere elegir la opción local, la aplicación mostrará un enlace desde dónde el usuario podrá guardar la copia de seguridad en la carpeta local que desee. Para realizar la copia de seguridad sólo se debe pulsar en el botón «Realizar backup», véase Figura 24.

Después de hacar clic en el botón aparecerá el enlace para descargar la copia de seguridad recién realizada, véase Figura 25

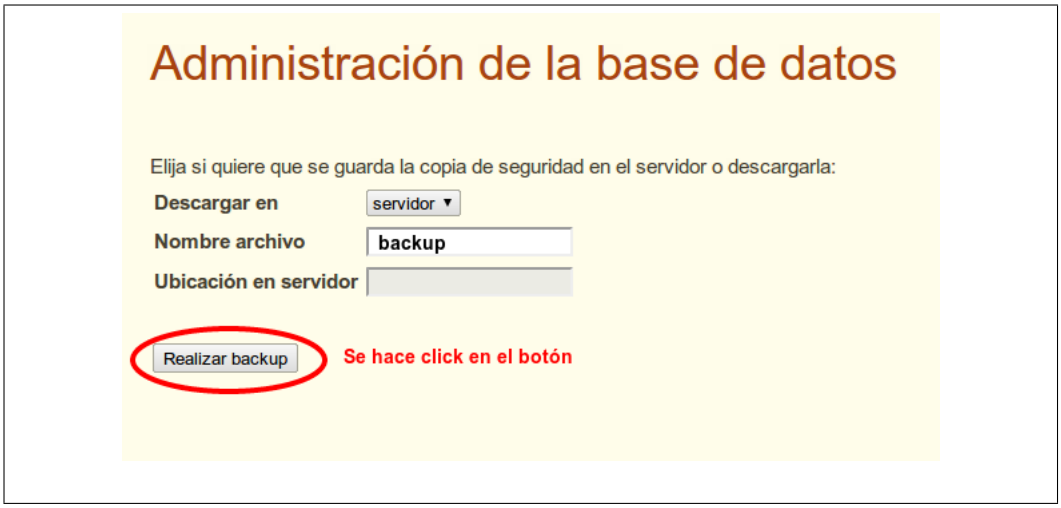


Figura 24: *Primer formulario para realizar una copia de seguridad de la base de datos*

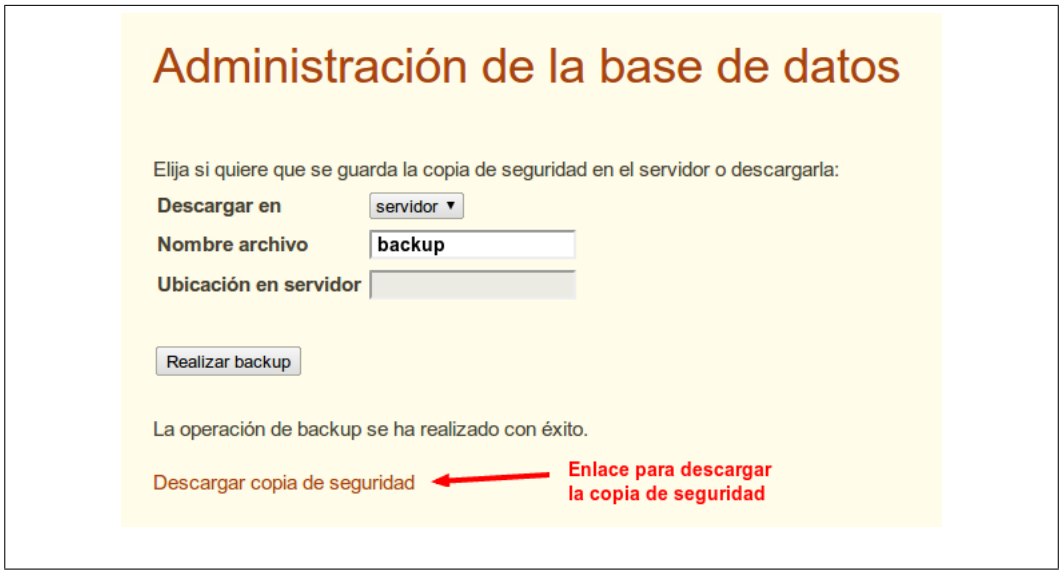


Figura 25: *Segundo formulario para realizar una copia de seguridad de la base de datos*

.3.13.6. Cargar copia de seguridad

Esta operación permite cargar en la base de datos, una copia de seguridad que existiera previamente. La copia de seguridad se puede cargar desde un fichero existente en el servidor o bien desde un fichero local que previamente subiremos a la aplicación. Si la carga de la copia de seguridad se realiza desde un fichero alojado en el servidor de la aplicación, es necesario facilitar la ubicación exacta del mismo. Para ello, se debe escribir la ubicación del fichero como se muestra en la siguiente imagen. **Importante: la ubicación del fichero debe tener la extensión del mismo**. Una vez escrita la ruta correcta, se pulsa el botón «Cargar backup» y el sistema realizará la carga del backup, véase Figura 26

Figura 26: *Formulario de carga de una copia de seguridad de la base de datos alojada en el servidor*

Si la carga se realiza desde un fichero local, entonces es necesario subir previamente el fichero a la aplicación. Para ello, hay que hacer clic en el botón «Elegir un archivo». Seguidamente, aparecerá una ventana dónde elegiremos el archivo dentro del sistema de ficheros de nuestro sistema operativo. Una vez elegido el fichero, se pulsa el botón «Cargar backup» y el sistema realizará la carga del backup. Véase Figura 27.

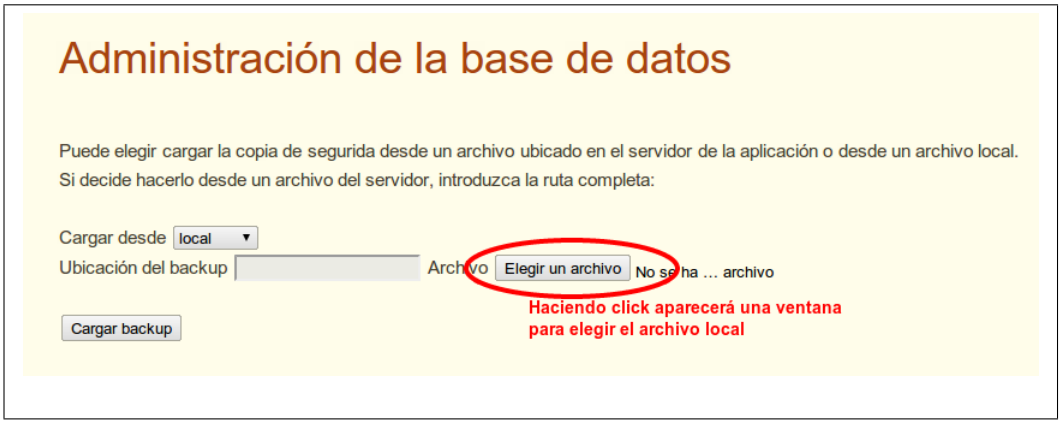


Figura 27: *Formulario de carga local de una copia de seguridad de la base de datos*